

⑨ 日本国特許庁(JP)

⑩ 特許出願公開

⑫ 公開特許公報(A) 平4-153733

⑬ Int. Cl.³

G 06 F 9/38

識別記号

3 1 0 E
3 1 0 X
3 3 0 X
3 7 0 B

庁内整理番号

7927-5B
7927-5B
7927-5B
7927-5B

⑭ 公開 平成4年(1992)5月27日

審査請求 未請求 請求項の数 1 (全36頁)

⑮ 発明の名称 並列処理装置における命令供給装置

⑯ 特 願 平2-279654

⑰ 出 願 平2(1990)10月17日

⑱ 発 明 者 安 藤 秀 樹 兵庫県伊丹市瑞原4丁目1番地 三菱電機株式会社エル・エス・アイ研究所内

⑲ 出 願 人 三菱電機株式会社 東京都千代田区丸の内2丁目2番3号

⑳ 代 理 人 弁理士 深見 久郎 外2名

明 細 書

1. 発明の名称

並列処理装置における命令供給装置

2. 特許請求の範囲

各々が所定の機能を実行する複数の機能ユニットと、命令を格納する命令メモリ装置と、命令フェッチ指示にตอบสนองして前記命令メモリ装置から複数の命令を同時にフェッチする命令フェッチ装置と、該フェッチされた複数の命令から同時に実行可能な命令を見出し、該同時実行可能な命令に関連の機能ユニットへ発行する命令デコード装置とを含む並列処理装置における命令供給装置であって、前記命令デコード装置は、前記関連の機能ユニットへ発行した命令数を示す情報と、与えられた命令に分岐命令が含まれておりこの分岐命令により分岐が発生したことを示す分岐発生指示情報とを発生する手段を含み、

前記命令メモリ装置からの命令の供給待ち状態であるか否かを示す命令供給有無指示信号と、リセット信号と、前記分岐発生指示情報とにตอบสนองし

て、命令の有効・無効を示す有効性フラグを形成する手段、

前記同時にフェッチされた複数の命令と、該同時にフェッチされた複数の命令の前記命令メモリ装置におけるアドレスと、前記有効性フラグとを同時に格納する記憶手段、前記記憶手段は、1エントリが1命令と対応のアドレスと対応の有効性フラグとからなる複数のエントリ格納領域を備えかつ、各々に異なる書込エントリが同時に伝達される複数の入力ポートおよび前記複数の入力ポートと別に設けられ、各々に異なる読出エントリが同時に伝達される複数の出力ポートと、書込キューアドレスおよび読出キューアドレスにそれぞれตอบสนองして複数のエントリを前記複数の入力ポートおよび前記複数の出力ポートへ同時にそれぞれ接続する手段とを含み、および

前記命令発行数指示情報、前記分岐発生指示情報、前記命令供給有無指示信号、および前記リセット指示信号にตอบสนองして、前記書込キューアドレスおよび前記読出キューアドレスをそれぞれ生成

して前記記憶手段へ与えるアドレス生成手段を備え、

前記記憶手段から同時に読出された複数のエントリの内容が前記命令デコード装置へ与えられる、並列処理装置における命令供給装置。

3. 発明の詳細な説明

【産業上の利用分野】

この発明は並列処理装置に関し、特に、スーパー scaler 型プロセサの命令供給方式の効率化に関する。

【従来の技術】

近年のマイクロプロセサの進歩には目覚ましいものがあり、その高性能化とともに動作速度も高速化されてきている。一方、半導体メモリの高速化がこのマイクロプロセサの高速化に追隨することができず、半導体メモリのアクセスがプロセサの高速化に対するボトルネックとなっている。このため、並列処理を行なうことによりマイクロプロセサの性能向上を図ることが行なわれている。

このような並列処理を実現する方式の1つに、

ユニット4、5、6および7と、命令を格納する命令メモリ1と、命令メモリ1から複数の命令を同時にフェッチする命令フェッチ(IF)ステージ2と、命令フェッチステージ2によりフェッチされた複数の命令を同時に受け、この中から同時に実行処理可能な命令を見出して対応の機能ユニットへ投入する命令デコード(ID)ステージ3と、演算処理結果等を格納するためのデータメモリ8を含む。命令メモリ1は、キャッシュメモリおよび外部メモリを含み、キャッシュヒット(キャッシュメモリ内に要求されたメモリが存在する状態)の場合には高速で命令を読出すことができる。

命令フェッチステージ2は、命令メモリ2へ命令ポインタIF_PCを与え、この命令ポインタIF_PCに対応する複数の命令を命令メモリ1からフェッチする。

命令デコードステージ3は、命令デコードおよびパイプラインシーケンサを含む。命令デコードは命令フェッチステージ2によりフェッチされた複数の命令を受けて解釈する。パイプラインシー

ケンス(命令スケジューラ)と呼ばれる処理方式がある。このスーパー scaler 型のプロセサ(以下、単にスーパー scaler と称す)は、第30図に示すように、スーパー scaler 内部のスケジューラ200が、命令ストリーム中の並列性を検出し、並列処理可能な命令を並列に設けられたパイプラインP1、P2およびP3へ供給する方式である。すなわち、スーパー scaler は以下の特徴をもつ処理装置であるといえる。

(1) 命令を複数個同時にフェッチする。

(2) 複数の機能ユニット(パイプライン)を有しており、複数の命令を同時に実行することができる。

(3) フェッチされた複数の命令の中から同時に実行可能な命令を見つけ出し、この同時に実行可能な命令に関連の機能ユニット(パイプライン)へ同時に投入する。

第31図はスーパー scaler の一般的構成を示す図である。第31図において、スーパー scaler は、各々が所定の機能を実行する複数の機能ユ

ニット(命令スケジューラ)はこのデコードされた複数の命令のたとえばマシンタイプを識別し、異なるマシンタイプの命令を同時に対応の機能ユニットへ投入する。ここで、マシンタイプとは、命令がどの機能ユニットで処理されるべきかを示す情報である。

機能ユニット4ないし7は、それぞれパイプライン化されており、与えられた命令をクロック信号にตอบสนองして実行する。この第31図においては、4つの機能ユニットが一例として示されており、最大4つの命令が並列処理可能である。機能ユニット4および5は整数加算などを行なうための整数演算ユニットであり、整数演算を行なう実行ステージ(EX)および書込ステージ(WB)を含む。書込ステージ(WB)は、処理の実行結果をデータレジスタ(図示せず)へ書込む。

機能ユニット6は、データメモリ8へのアクセス(データのロードまたはストア)を実行するユニットであり、アドレス生成ステージ(ADR)、データメモリ8へのアクセス実行ステージ(ME

M) およびデータレジスタ(図示せず)へのデータの書込ステージ(WB)である。ここで書込ステージ(WB)においてはデータのロードまたはデータのストアが行なわれる。

機能ユニット7は、浮動小数点加算などを実行するためのユニットであり、3段の実行ステージ(EX1、EX2およびEX3)と、実行結果のデータレジスタへの書込を行なう書込ステージ(WB)を含む。浮動小数点数は指数と仮数とを用いて表わされ、その小数点の位置が一定しない数であり、浮動小数点演算は整数演算などに比べてその実行により多くのサイクルが必要とされる。

この並列処理装置においては、各段はすべてパイプライン化されており、命令フェッチ、命令デコードおよび命令実行ならびにデータ書込が互いにオーバーラップしつつ実行される。したがって、命令メモリからフェッチされた命令が次のサイクルで命令デコードステージでデコードされる。次に動作について簡単に説明する。

ト4ないし7)の動作はパイプライン化されており、それぞれ互いにオーバーラップしながら所定の動作を実行している。

上述のように各ステージの動作をパイプライン化し、かつ複数の機能ユニットで処理を並列に実行することにより高速で命令を実行することができる。

[発明が解決しようとする課題]

上述のように、スーパー scaler においては、複数の命令がフェッチされ、かつ同時に複数の命令が実行されるため、通常のコンピュータに比べて処理速度の向上を図ることができる。たとえば、第31図に示す構成において、同時にフェッチされた4つの命令が4つの機能ユニット4ないし7で並列に実行された場合、4クロックサイクルで4命令を処理することができる(機能ユニット4、5および6のパイプラインが機能ユニット7の処理完了まで待ち合わせ状態となる場合)。

命令スケジューラ(または命令デコードステージに含まれるパイプラインシーケンサ)は、効率

命令デコードステージ3は、命令フェッチステージ2へ命令フェッチ要求を出力する。命令フェッチステージ2は、この命令フェッチ要求に応じて命令ポインタIF_PCを命令メモリ1へ与え、命令メモリ1からその命令ポインタIF_PCに対応する複数の命令をフェッチする。このフェッチされた複数の命令は命令デコードステージ3へ同時に与えられる。命令デコードステージ3はこの与えられた複数の命令を同時にデコードする。この命令デコードステージ3はさらに、デコードされた命令から、計算リソースおよびデータレジスタが飽和しない、並列処理可能な命令を検出し、この並列処理可能な命令を対応の機能ユニットへ発行(投入)する。命令を与えられた機能ユニットはこの与えられた命令に従って処理を並列に実行する。機能ユニット4ないし7の処理の実行は、パイプライン化されており、この第31図に示す各ステージに従って処理が実行される。また、命令フェッチステージ2および命令デコードステージおよび命令実行ステージ(機能ユニッ

的に並列処理が実行されるように命令のスケジューリングを実行するが、同時にフェッチされた命令が常に同時に処理されるとは限らない。たとえば、以下のようなデータ依存関係がある命令を考える。

① add R1, R2, R3

; R2 + R3 = R1

② sub R4, R1, R5

; R1 - R5 = R4

命令①はレジスタR2の内容にレジスタR3の内容を加え、その加算結果をレジスタR1に書込む命令である。

命令②はレジスタR1の内容からレジスタR5の内容を減算し、この減算結果をレジスタR4に書込む命令である。このような命令①および②の演算はたとえば「x+y-z」という処理に対応する。

この命令①および②が同時にフェッチされたとき、これらはレジスタR1を共通に使用しており、命令②は命令①の結果を使用することになるため、

これらの命令①および②は同時に実行することができない。このような命令間のデータ依存関係が存在する場合、命令の発行状況（命令デコードステージから機能ユニットへの命令の発行態様）は第32図に示すようになる。

第32図においては、同時に処理可能な命令のみが発行される状況が示されており、また同時に発行される命令は、アドレスの小さい方（図の左側に位置する）の命令から順に決定される。

第32図において、サイクル1においては、命令2、3および4が相互にまたは命令1に対しデータ依存関係を有するため、これらは発行することができず、命令1のみが発行される。

サイクル2においては、命令4は命令2および/または命令3とデータ依存関係があり、命令2および命令3は互いにデータ依存関係を有しないため、命令2および3が発行される。

サイクル3においては、残った命令4が発行される。

サイクル4において、新たにフェッチされた4

つの命令5ないし8のデコードが行なわれ、相互にデータ依存関係のない命令5および6が発行される。

サイクル5において命令7および8が相互にデータ依存関係を有しないためこれらが発行される。ここで、第32図においては□印で囲んだ数字は、相互にデータ依存関係のない発行可能な命令を示している。

この第32図に示すような、同時にフェッチされた複数の命令がすべて発行されるまで次の命令のフェッチを待ち合わせる方式の場合、最初にフェッチされた命令1ないし4と、次にフェッチされた命令5ないし8がすべて発行されるまでに5サイクルが必要とされる。したがって、このような命令供給および発行方式では、パイプラインに空きが生じ、並列処理装置の高速処理性能が損なわれる。

このような場合もし、第32図において命令4と命令5および6との間にデータ依存関係が存在せず、かつ任意の数の命令を命令メモリからフェ

ッチすることが可能であれば、第33図に示す命令発行方式を実現することができる。第33図は改良された命令発行方式における命令発行状況を示す図である。

第33図を参照して、サイクル1において、同時にフェッチされた4つの命令1ないし4のうち、命令1が発行される。

サイクル2において、新たに命令5が供給され、命令2ないし5のデコードが行なわれる。この4つの命令2ないし5のうち、依存関係のない命令2および3が発行される。

サイクル3において、新たに命令6および7が供給され、命令4ないし7がデコードされる。このデコード結果に従って命令4、5および6が発行される。

サイクル4において、新たに命令8、9および10が供給され、命令7ないし10のデコードが行なわれ、このデコード結果に従って命令7および8が発行される。

この第33図に示す命令発行方式においては、

命令1ないし8の8個の命令の発行に4サイクルが必要とされるだけであり、第32図に示す方式に比べてより高速で命令の処理実行を行なうことができる。この第33図に示す命令発行手順を実現する手法の1つとして第34図に示す方法が考えられる。第34図は第33図に示す命令発行状況を実現するための命令の供給手順を示す図である。

第34図を参照して、ステップ（1）において、命令レジスタ（命令を保持するレジスタ）に保持された命令2ないし5のうち命令2および3が発行されると命令レジスタに空きが生じる。

ステップ（2）において、この空きの生じたレジスタの数だけ命令レジスタの内容をシフトさせる。すなわち第34図のステップ（2）において命令4および5は左へ2つだけレジスタ位置がシフトされる。

ステップ（3）において、この空きの生じた命令レジスタ内へ次の命令6および7がフェッチされる。この第34図に示すステップ（1）ないし

(3)を1サイクルで実行する必要がある。この第34図に示す命令シフト動作を実行する構成としては第35図に示すような構成を考えることができる。

第35図において、命令デコードステージは、命令を格納する命令レジスタIR1～IR8と、与えられた命令をシフトさせるバレルシフトBRと、命令デコードIDを含む。命令レジスタIR1～IR4は命令フェッチステージ2によりフェッチされた命令およびバレルシフトからシフトされた命令を格納する。命令レジスタIR5ないしIR8は、命令デコードIDへ与えられた命令がデコードされずに格納される。バレルシフトBRは、命令レジスタIR5ないしIR8からの命令のうち、命令デコードIDからの発行命令数情報に従って命令をシフトさせる。

通常、並列演算処理装置は第36図に示すように2相の互いに重なり合わないクロック信号TおよびLに従って動作している。次に動作について簡単に説明する。

を実行することができず、命令シフト動作に長時間を要する。したがって、次のクロック信号Tにตอบสนองして命令レジスタIR1～IR4へ命令を与えて保持させることができなくなるため、高速で命令デコードIDへ命令を供給することができなくなり、並列処理装置の高速動作性が損なわれる。

また、命令メモリは、この命令レジスタIR1～IR4においていくら空きレジスタが存在しているかという情報を知り、この情報に基づいて命令供給数および命令供給位置を定める必要がある。この命令レジスタIR1～IR4においていくつ空きレジスタが存在するかの情報は命令デコードからの発行命令数情報で得られるが、この命令発行数情報が発生されるのは命令デコードIDにおけるデコード動作後であり、また、この情報に従って、命令メモリにおいて命令供給数および命令供給位置を判定するのに長時間を要し、またその判定動作開始タイミングも遅い。このため、所望の命令をフェッチするのに長時間を要することになり、パイプライン構成を乱さずに実行するため

前のサイクルでフェッチされた命令がクロック信号Tにตอบสนองして命令レジスタIR1～IR4で保持される。命令レジスタIR1～IR4の保持命令は命令デコードIDへ与えられ、デコードされ、該デコード結果に従って機能ユニットへ発行される。

一方、命令レジスタIR5～IR8はクロック信号Lにตอบสนองしてその命令デコードIDへ与えられた命令を保持しバレルシフトBRへ与える。バレルシフトBRは、命令デコードIDからの発行命令数情報にตอบสนองして、この命令レジスタIR5～IR8から与えられた命令をシフトさせる。このバレルシフトBRの内容は、次のクロック信号Tにตอบสนองして命令レジスタIR1～IR4へ与えられて保持される。

この場合、バレルシフトはクロック信号Lから次のクロック信号Tの間にその命令のシフト動作を完了していなければならない。しかしながら、バレルシフトBRにおいては、命令長が長く（たとえば32ビット）、高速でその命令シフト動作

には、クロック信号TおよびLの周期が長くなり、並列演算処理装置の高速動作性が損なわれる。

それゆえ、この発明の目的は並列処理を効率的に行なうことのできる並列処理装置を提供することである。

この発明の他の目的は並列処理を効率的に行なうことのできる命令供給装置を備えた並列処理装置を提供することである。

【課題を解決するための手段】

この発明に係る並列処理装置における命令供給装置は、命令メモリ装置からの命令の供給の有無を示す信号と、リセット信号と、分岐が発生したことを示す分岐発生指示情報とにตอบสนองして、命令の有効／無効を示す有効性フラグを形成する手段と、命令メモリから同時にフェッチされた複数の命令と、この同時にフェッチされた複数の命令の上記命令メモリにおけるアドレスと有効性フラグとを同時に格納する記憶手段を含む。この記憶手段は、1エントリが1命令と、対応のアドレスと、対応の有効性フラグとからなる複数のエントリ格

納領域を備える。またこの記憶手段は、各々に異なる書込エントリ内容が同時に伝達される複数の入力ポートと、各々に異なる読出エントリ内容が同時に伝達される複数の出力ポートと、書込キューアドレスおよび読出キューアドレスにตอบสนองして複数の書込エントリおよび複数の読出エントリを複数の入力ポートおよび複数の出力ポートへそれぞれ接続する手段を含む。

この命令供給装置はさらに命令デコードからの命令発行数情報と分岐発生指示情報と命令供給有無指示信号とにตอบสนองして、上記書込キューアドレスおよび読出キューアドレスをそれぞれ生成する手段を備える。

この記憶手段から同時に読出された複数のエントリの内容が命令デコード装置へ与えられる。

【作用】

記憶装置は、互いに独立にアクセス可能な複数の入力ポートおよび複数の出力ポートを含む。この複数の入力ポートへは同時に命令メモリからフェッチされた複数の命令、アドレスおよび有効性

フラグを伴ってそれぞれの入力ポートを介して異なるエントリ領域へ同時に格納される。

またこの複数のエントリ格納領域からは読出キューアドレスにตอบสนองして同時に複数のエントリ領域が選択されそれぞれ異なる出力ポートに接続され、これにより複数のエントリ内容が同時に読出される。

この入力ポートおよび出力ポートはそれぞれ独立にアクセス可能であるため、エントリへの書込読出を同時に実行することができる。また書込キューアドレスおよび読出キューアドレスは、それぞれ命令の発行状況および命令メモリからの命令供給状況に応じて選択され、常に同一数の命令が発行対象命令として命令デコード装置へ与えられる。有効性フラグは該命令の発行の可否を決定する。

これにより、命令デコード装置へは、常に発行された命令数を補償する命令が同時に与えられ、効率的に命令供給を実行することができることになり、命令実行を高速で行なうことが可能となる。

並列処理装置を実現する。

【発明の実施例】

第1図はこの発明の一実施例であるスーパーカラーの要部の構成を示す図である。第1図に示す構成はスーパーカラーの命令デコード・ステージに対応する。この第1図に示す装置は、2相の互いに重なり合わないクロック信号TおよびL（第36図参照）にตอบสนองして動作する。

図示しないが命令フェッチ・ステージ（第36図参照）は、この命令デコード・ステージからの命令フェッチ要求に従って複数（本実施例では4つ）の命令を命令メモリからフェッチする。フェッチした命令IC_dataおよびこのフェッチした命令の論理アドレスIF_PC_Lは、命令が命令メモリから供給されることを示す信号IC_readyとともにこのデコード・ステージへ供給される。フェッチ命令の論理アドレスIF_PC_Lは、命令フェッチ・ステージに含まれる、命令メモリのアドレスを作成するポインタから供給される。

命令メモリは、たとえばキャッシュミス等により低速の主メモリ（または外部メモリ）へアクセスしている状態または命令メモリがアクセスを受けている状態等においては、命令メモリが命令フェッチ要求を受入れられないことを示す信号IC_busyを発生してデコード・ステージへ与える。

第1図を参照して、この発明の一実施例である命令供給装置は、命令メモリからフェッチされた命令の論理アドレス（IF_PC_L）、フェッチされた命令（IC_data）およびこの命令の有効／無効を示す有効性フラグ（inst_avail）を1エントリとして格納するランダム・アクセス・メモリ（RAM）からなるキュー9を含む。このキュー9は、後にその構造を詳細に説明するが、たとえば12エントリを格納することができ、かつ複数の書込ポートおよび複数の読出ポートを有する。この複数の書込ポートおよび複数の読出ポートにより、1つの書込ゲートおよび読出ゲートを指定することにより、連続する複

数（本実施例では4つ）のエントリの内容を同時に書込または読出すことができる。

このキュー9への命令の書込および読出ならびに格納された命令の有効無効を制御するために、クロック信号Tにตอบสนองして、与えられた信号を保持するTラッチ回路10、11、12、13および28と、各Tラッチ回路10～13および28に対応して設けられ、対応のTラッチ回路の出力データ（保持データ）をクロック信号Lにตอบสนองして保持するLラッチ回路17、18、19、20および29と、各Lラッチ回路17～20および29の保持データと命令デコード26からの信号br_taken、issued_inst_count、およびbranch_inst_entryとから、キュー9の読出アドレス、書込アドレス、動作制御信号および命令の有効／無効を示す信号を発生する論理回路30、31、32、33、34、35および36ならびに50を含む。

Tラッチ回路10（以下、キュートップラッチ10と称す）は、キュー9に命令メモリからフェ

9のアドレス：

scope, scope+1,
scope+2, scope+3

のエントリの内容が読出される。ここで、「+」はキュー9の全エントリ数（本実施例では12）についてのモジュロ加算である。

Tラッチ回路（以下、命令フェッチラッチと称す）12は、命令フェッチを行なうことを示すフラグIC_fetchをクロック信号Tにตอบสนองして保持する。この命令フェッチ要求フラグIC_fetchが“1”のサイクルにおいては、命令フェッチ要求が命令フェッチステージへ発行される。

Lラッチ回路（以下、命令フェッチLラッチと称す）19は、クロック信号Lにตอบสนองして、命令フェッチラッチ12にラッチされたフラグIC_fetchを保持する。

Tラッチ回路（以下、命令ミスラッチと称す）13は、命令メモリからの命令の待機状態であることを示すフラグIC_missをクロック信号

ラッチされた命令を書込むべき複数のエントリの最初のアドレスを示すアドレスqueue_topをクロック信号Tにตอบสนองして保持する。すなわち、キュー9に4つの命令が格納される場合、命令、その論理アドレスおよび有効性フラグはキュー9のアドレス：

queue_top, queue_top+1
queue_top+2, queue_top+3

に書込まれる。ここで「+」はキュー9が有する全エントリ数（本実施例では12）についてのモジュロ加算である。

Lラッチ回路17（以下、キュートップLラッチ17と称す）は、キュートップラッチ10の保持データを、クロック信号Lにตอบสนองして保持する。

Tラッチ回路11（以下、スコブラッチ11と称す）は、クロック信号Tにตอบสนองして、キュー9から同時に読出されるべき4エントリの最初の1エントリが登録されているキュー9のアドレスをクロック信号Tで保持する。すなわち、キュー

Tにตอบสนองして保持する。

Lラッチ回路（以下、命令ミスLラッチと称す）20は、この命令ミスラッチ13の保持するフラグIC_missをクロック信号Lにตอบสนองして保持する。

Tラッチ回路（以下、初期状態ラッチと称す）28は、クロック信号Tにตอบสนองしてキュー9の初期状態を表わすフラグqueue_init_stateを保持する。

Lラッチ回路（以下、キュー初期状態Lラッチと称す）29は、キュー初期状態ラッチ28の保持するフラグqueue_init_stateをクロック信号Lにตอบสนองして保持する。

論理回路30（以下、キュートップロジック30と称す）は、命令デコード26からの分岐発生信号br_takenと、キュー初期状態Lラッチ29の保持するキュー初期状態フラグqueue_init_stateと、信号IC_readyと、キュートップLラッチ17からのキュー9の書込先頭アドレスqueue_topと、命

令フェッチラッチ19からのフラグIC_fetchと、論理回路31からの次のサイクルにおけるキュー9の読出先頭アドレスscope_nextと、キュー9が利用可能(書込可能)であるか否かを示す論理回路34からのフラグqueue_availおよび命令デコード26から発生される、分岐命令が登録されたエントリのアドレスbranch_inst_entryとに回答して、次のサイクルにおけるキュー9の書込先頭アドレスqueue_top_inを形成し、キュートップラッチ10、論理回路34および50へ与える。

論理回路31(以下、スコープネクストロジック31と称す)は信号br_takenと、スコープラッチ18からの読出先頭アドレスscope_Lと、命令デコード26からの発行命令数データissued_inst_countおよびアドレスbranch_inst_entryとに回答して、次のサイクルにおけるキュー9の書込先頭アドレスscope_nextを発生す

ェッチラッチ19からの信号IC_fetch_Lと、信号IC_readyとに回答してキュー9が初期状態であることを示す信号queue_init_stateを作成してキュー初期状態ラッチ28へ与える。

論理回路34(以下、キューアベールロジック34と称す)は、キュートップラッチ17からのアドレスqueue_top_Lと、スコープネクストロジック31からのアドレスscope_nextと、信号br_takenと、キュートップロジック30からのアドレスqueue_top_in、論理回路50からのキュー9の状態(第1、第2および第3の状態:これは後に説明する)を示す信号queue_stateとに回答して、キュー9が利用可能であるか否かを示す信号queue_availを発生し、キュートップロジック30へ与える。

論理回路32(以下、命令フェッチロジック32と称す)は、命令メモリまたは命令フェッチステージから与えられる、命令メモリが命令フェ

るとともに、このアドレスscope_nextが11より大きいことを示すフラグscope>11を発生する。次のサイクルにおけるアドレスscopeを示す信号scope_nextはスコーブラッチ11および論理回路50へ与えられるとともに論理回路34へも与えられる。フラグscope>11は論理回路50へ与えられる。

論理回路33(以下、命令ミスロジック33と称す)は、命令フェッチラッチ19からのフラグIC_fetch_Lと、命令ミスラッチ20からのフラグIC_miss_Lと、フラグbr_takenと、信号IC_readyとに回答して、命令待ち状態であるか否かを示す信号IC_missを作成して命令ミスラッチ13へ与える。

論理回路36(以下、キュー初期状態ロジック36と称す)は、キュー初期状態ラッチ29からの信号queue_init_state_Lと、信号br_takenと、命令ミスラッチ20からの信号IC_miss_Lと、命令フ

チ要求を受入れられない状態を示す信号IC_busyに回答して、命令メモリへフェッチ要求を出すか否かを示す信号IC_fetchを発生し、命令フェッチラッチ12へ与える。

論理回路35(以下、命令アベールロジック35と称す)は、信号IC_readyと、信号IC_fetch_Lと、信号br_takenと、信号IC_miss_Lとに回答して、キュー9に含まれる命令が有効であるか否かを示すフラグinst_avail_inを発生し、キュー9のエントリ内の有効性フラグinst_availの値を決定する。

論理回路50(以下、キューステートロジック50と称す)は、命令デコード26からのアドレスbranch_inst_entryと、キュートップロジック30からのフラグqueue_top>11と、スコープネクストロジック31からのフラグscope>11とアドレスscope_nextとに回答して、キュー9が、第1の状態、第2の状態および第3の状態のいずれに

あるかを示す信号 `queue_state` を発生し、キューアペールロジック34へ与える。ここで第1の状態はキュー9が初期状態にあり、その書込先頭アドレス `queue_top` および読出先頭アドレス `scope` がともに0または同一であり、またアドレス `queue_top` からの4エントリの有効性フラグ `inst_avail` がすべてオフ(0)の状態を示す。この状態はリセット状態、分岐発生状態および命令到着待ち状態のいずれかの状態である。第2の状態は、読出先頭アドレス `scope` が書込先頭アドレス `queue_top` 以下の状態、すなわち、

$$scope \leq queue_top$$

の状態でありノーマル(NORMAL)状態である。

第3の状態は書込先頭アドレス `queue_top` が読出先頭アドレス `scope` 以下の場合であり、すなわち、

$$queue_top \leq scope$$

の状態を示す。この第3の状態はリバース(RE

VERSE) 状態と呼ばれる。この第2の状態および第3の状態を区別することにより、アドレス `scope_next` とアドレス `queue_top_in` が等しくなった状態において、キュー9が空の状態であるのか満杯(フル)状態であるのかを区別することができる。すなわち、第2の状態から `scope_next = queue_top_in` となった場合、このときはキュー9は空の状態を示している。また、第3の状態から `scope_next = queue_top_in` となった場合、このときはキュー9はフル状態となる。この第2および第3の状態については後に詳細に説明する。

キュー9と命令デコード26との間には、キュー9から読出された複数のエントリの内容をクロック信号Lにตอบสนองして保持するLラッチ回路14、15および16が設けられる。Lラッチ回路14(以下、IRLラッチ14と称す)は、キュー9から同時に読出された複数の命令 `inst` を保持する。

Lラッチ回路15(以下、PCLラッチ15と称す)は、キュー9から同時に入力された命令に対応するアドレスPCを保持する。

Lラッチ回路16(以下、命令アペールLラッチ16と称す)は、キュー9から読出された命令に付随する有効性フラグ `inst_avail` を保持する。

命令デコード26は、このIRLラッチ14、PCLラッチ15および命令アペールLラッチ16からのアドレス、命令および有効性フラグにตอบสนองして、与えられた複数の命令をデコードし、並列実行可能な命令を検出し、その命令を機能ユニットへ発行する。この同時に機能ユニットへ発行される命令は、PCLラッチ15からのアドレスの小さい順に決定され、複数の命令が依存関係を有する場合には、アドレスの小さい方の命令が先に発行される。

次に、この発明による命令フェッチ方式の基本的な動きについて説明する。

第2図は、キュー9の概念的構成を示す図であ

る。第2図において、キュー9は、アドレス0ないし11の付された記憶領域を有し、各アドレスに対応して1個のエントリEが格納される。各エントリEは、命令メモリからフェッチされた命令のアドレス(論理アドレス)を格納する領域Iと、命令メモリからフェッチされた命令 `IC_data` を格納する領域IIと、この対応の命令が有効であるか否かを示すフラグ `inst_avail` を格納する領域IIIを含む。このアドレス `IF_PC_L`、命令 `IC_data` およびフラグ `inst_avail_in` はそれぞれ4つの命令が並列にこのキュー9に伝達され、4つの異なるアドレス領域に同時に格納される。このキュー9のエントリの登録領域はアドレス `queue_top` により指定され、アドレス `queue_top ~ queue_top + 3` の領域に格納される。

このキュー9からの命令の読出は、4つの命令が並列に読出されることにより行なわれる。この読出のアドレスの指定は、アドレス `scope` により行なわれ、アドレス `scope ~ scope`

+3のアドレスに登録されたエントリが並列に同時に読出される。次に、第3図を参照してこの発明による命令フェッチ方式の基本的な動作について説明する。

まず状態(1)はリセット直後または初期化直後の初期状態(第1の状態)である。この初期状態においては、アドレス`queue_top`および`scope`はともにキュー9の初期アドレス0を示している。このとき、キュー9においては`queue_top`すなわちアドレス0から始まる有効性フラグはオフ(0)とされる。この状態においては、有効性フラグがオフ(0)を示しているためそのキュー9のアドレス0~3の4エントリは無効であり、発行または実行が禁止されていることを示している。

状態(2)において、アドレス`queue_top`が示すアドレス0から始まる4つのアドレスに対して命令が書込まれる。この書込まれた命令は有効な命令であるため、これに対応する有効性フラグがオン(1)とされ、この書込まれた命令

3を示す。またこの状態(4)においては、新たに4つの命令8ないし11が書込先頭アドレス`queue_top`が示すアドレス8から連続する4つのアドレス領域にそれぞれ書込まれており、対応の有効性フラグもオン(1)とされる。この状態(4)のサイクルにおいては、アドレス`scope`が示すアドレス3から始まる4つのアドレスの命令がキュー9から読出され命令デコーダへ与えられて解読される。

状態(5)は、前のサイクルにおいて命令デコーダからは1つも命令が発行されなかった場合の状態を示している。この場合、発行された命令数は0であるため、書込先頭アドレス`scope`は移動せず、アドレス3を示す。この状態(5)のサイクルにおいては、アドレス`scope`が示すアドレス3から始まる4つのアドレスの命令が再びキュー9から読出されて命令デコーダにより解読される。この状態(5)においては、先のサイクルにおいて`queue_top+4`(12のモジュロ加算)が行なわれ、書込先頭アドレス`qu`

が有効であることが示され、この4つのエントリの命令の読出、発行/実行可能なことが示される。この状態(2)においては、まだ`queue_top`の変更は行なわれていない。この状態(2)においては、アドレス`scope`が示すアドレス0から始まる4つのエントリが読出され、命令デコーダへ与えられ、解読される。

状態(2)において、命令デコーダから2つの命令が同時処理可能であるとして機能ユニットへ発行され、読出先頭アドレス`scope`がアドレス2へ変更される。

状態(3)のサイクルにおいては、書込先頭アドレス`queue_top`がアドレス4に変更されており、このアドレス4から始まる4つのアドレス領域に次の4つの命令4ないし7がそれぞれ格納される。この新たに書込まれた命令4ないし7に対応する有効性フラグも同時にオンとされる。

状態(4)は、1つの命令が発行された後の状態を示している。1命令が発行されたため、読出先頭アドレス`scope`は1つ移動し、アドレス

`queue_top`は0($8+4=0$)となり、アドレス0を示している。この場合、`queue_top`から始まる4つのアドレス領域は、まだ発行されていない命令(アドレス3の命令)を含んでいるため、キュー9への命令の取込は行なわれない。

状態(6)は、前のサイクルで命令デコーダから3つの命令が発行された場合の状態を示している。3命令が発行されたため、読出先頭アドレス`scope`は3シフトされ、アドレス6を示している。書込先頭アドレス`queue_top`が示すアドレス0から始まる4つのアドレス領域に命令12、13、14および15が格納される。この命令12ないし15は有効であるため、それに対応する有効性フラグがオン状態とされる。このキュー9において空き領域があるか否かの判定は、キューアベールロジック34から出力される信号`queue_avail`により行なわれる。

次に、分枝命令による分枝が生じた場合の基本的動作についてそのフローを示す第4図を参照し

て説明する。いまキュー9が状態(11)にあった場合を想定する。この状態(11)においては、アドレス`queue_top`はアドレス10を示しており、アドレス`scope`はアドレス2を示している。この状態(11)においては、アドレス10, 11, 0, 1に命令8, 9, 10および11が書込まれる。キュー9のアドレス2, 3, 4および5に格納されている命令2, 3, 4, 5が読出されて命令デコードへ与えられる。この状態(11)のサイクルにおいて、アドレス4の命令2が分岐命令であり、この分岐命令2に従って分岐が生じたとする。次の状態(12)においては以下のことが行なわれる。

キュー9のアドレス4以降の内容は、命令2において分岐が生じたためもはや不要である。

アドレス`queue_top`に続くアドレスすなわちアドレス4ないし7に格納された4エントリの有効性フラグをオフ(0)に設定する。アドレス`queue_top`および`scope`をキュー9のアドレス4に設定する。

`pe` (キューアドレス4) から始まる4エントリの命令20ないし23が命令デコードへ与えられ、解説される。

次に、プロセサからの命令フェッチ要求に対して、命令メモリから命令を供給することができない場合の基本的な動きについてそのフローを示す第5図を参照して説明する。

いま、キュー9が第5図に示す状態(21)にあった場合を想定する。この状態(21)においては、キュー9のアドレス0ないし3へ命令8ないし11が書込まれ、一方、キュー9のアドレス6, 7, 8および9に格納されている命令2, 3, 4および5が命令デコードへ与えられ、発行対象として解説される。

このサイクルにおいて命令2, 3および4が機能ユニットへ発行されたとする。

このサイクルにおいて、命令メモリに対して行なった命令フェッチ要求に対して、命令メモリは何らかの原因(たとえば命令メモリがキャッシュで構成される場合、キャッシュミス)によりこの

また、分岐先命令のアドレスは、このサイクルで命令メモリへ送られ、分岐先命令はその次のサイクルで命令メモリから供給される。すなわち、分岐先命令はこのサイクルではキュー9には書込まれない。

アドレス`scope`より始まる4エントリの命令は発行対象として解説することができる。しかしながら、これらの内容は機能ユニットへ発行してはならないものであるため、このサイクルにおいては、アドレス`queue_top`より始まる4エントリの有効性フラグがオフ(0)に設定される。

状態(13)においては、アドレス`queue_top`から始まるアドレス4ないし7の領域に分岐先命令20~23が命令メモリから供給されて書込まれる。またこの命令20ないし23は分岐先命令であり、解説・発行されるべきものであるため、その有効性フラグもオン(1)に設定される。

このサイクルにおいては、またアドレス`scope`

サイクル内で命令を供給できなかったとする。

状態(22)においては、前のサイクルで命令2, 3および4が発行されているため、アドレス`scope`が3つシフトしてアドレス9を示している。

また、アドレス`queue_top`はアドレス4を示している。命令メモリから命令が供給されていないため、このキュー9のアドレス4, 5, 6および7に格納されている命令はフェッチ要求が出された命令と異なるため、解説してはならない。そこで、命令が命令メモリから供給されない場合、このアドレス`queue_top`に続くアドレスに格納された4エントリの有効性フラグがオフ(0)に設定される。

状態(23)においてもまだ命令メモリから命令が供給されず、プロセサは命令到着待機状態にある。このためキュー9への命令書込は停止する。一方、命令フェッチが行なわれなくても、キュー9に未発行の命令が存在すれば命令デコードを介して機能ユニットへの命令発行が続行される。こ

の第5図に示す例においては、前のサイクル（状態（22））において2命令が発行されており、このアドレスscopeは11を示している。

状態（24）において、命令が命令メモリから到着したため、アドレスqueue_top（アドレス4）から始まるキュー9の4エントリに命令が書込まれる。すなわち、命令12～15がアドレス4～7にそれぞれ書込まれる。

また前のサイクルにおいて2つの命令（命令7および8）が発行されており、アドレスscopeはアドレス1を示す。

次に、第3図ないし第5図に示す動作を実現するためのロジック30ないし36の論理について説明する。まず、キュートップロジック30が実現するロジックについて説明する。

（1）初期状態：この状態は次の2つの状態を含む。

（a）リセット状態：信号reset=1

このとき、queue_top=0となる。

（b）初期化状態：信号queue_init

1かつIC_ready=0

このとき、アドレスqueue_topは前のサイクルと同じ値を維持するので、

queue_top_in=queue_top_L
となる。

（4）発行された命令フェッチ要求に対し命令供給が行なわれ、かつキュー9に空きが存在する場合

: IC_fetch_L=1かつIC_ready=1、かつqueue_avail=1

このとき、アドレスqueue_topは次のサイクルにおいて4シフトされるため、

queue_top_in=queue_top+4
となる。ここで、「+」はモジュロ12の加算である。したがって、

if (queue_top > 11)

{ queue_top_in = queue_top_L - 12 }

となる。

t_state_L=1

この状態は、キュー9の書込先頭アドレスはキュートップラッチ17の保持するアドレスに設定される。したがって、

queue_top_in=queue_top_L
となる。

次に非初期状態、すなわちreset・queue_init・state_L=0の場合について説明する。ここで「・」は論理積演算を示す。

（2）分岐発生状態：信号br_taken=1:

この状態においては、アドレスqueue_topは、キュー9において分岐命令が格納されたアドレスに設定される。したがって、このときは、
queue_top_in=branch_inst_entry
となる。

（3）命令フェッチ要求が出されても命令供給が行なわれない場合：IC_fetch_L=

（5）命令フェッチ要求が出され、命令供給が行なわれても、キューに空き領域（4エントリ分）が存在しない場合

: IC_fetch_L=1かつIC_ready=1かつqueue_avail=0

この状態はキュー9に4エントリ分の命令を書込む領域が存在しないため、命令の書込が行なわれない状態を示しており、アドレスqueue_topは前のサイクルと同じ状態を維持する。したがって、

queue_top_in=queue_top_L
となる。

（6）命令到着待ち状態の場合

: この状態は2つある。

（a）命令フェッチ要求が出されても何らかの原因（キャッシュミス等）により命令供給が行なわれない場合

: IC_fetch_L=1かつIC_ready=0

この状態では前のサイクルの`queue_top`が保持される。したがって、

`queue_top_in = queue_top_L`

となる。

(b) たとえば命令メモリがアクセスを受入れられない状態にあるとき

: この状態は`IC_miss_L = 1`かつ`IC_ready = 0`

この状態では、命令が供給されていないため、アドレス`queue_top`は前の値を保持する。したがって、

`queue_top_in = queue_top_L`

となる。

(7) たとえばキャッシュミス等による命令到着待ち状態において命令が与えられ、かつキュー9に空き領域が存在する場合:

`IC_miss_L = 1`かつ`IC_ready = 1`かつ`queue_avail = 1`

この状態はたとえば割込み処理などにより、命令メモリの命令の実行が中断される状態である。この状態ではアドレス`queue_top`は前のサイクルの値を維持する。したがって、

`queue_top_in = queue_top_L`

となる。この上述の論理を第6図に一覧にして示す。

このキュートップロジック30が実行する論理は上述の説明から明らかであろう。このキュートップロジック30の具体的構成は、第6図に示す論理表を満足する構成であればどのようなものであってもよい。たとえば単純には、この第6図に示す表の各列(横方向)が示すロジックをたとえばANDゲートにより形成し、この各列のロジックの論理和をとる回路構成を用いて実現できる。

次にスコープネクストロジック31の実行する論理動作について説明する。

(1) リセット状態: 信号`reset = 1`

この状態ではアドレス`scope`は0に設定さ

この状態では、与えられた命令をキュー9にすべて書込むことができる。このときには、このサイクルでは、`queue_top_in = queue_top_L + 4`

となる。ただし「+」はモジュール12のモジュロ加算である。

(8) 命令到着待ち状態において命令が供給されてもキュー9に空き領域が存在しない場合

: `IC_miss_L = 1`、かつ`IC_ready = 1`、かつ`queue_avail = 0`

このとき、キュー9には命令を書込むことができないので、アドレス`queue_top`は前のサイクルの値を保持する。したがって、

`queue_top_in = queue_top_L`

となる。

(8) 命令フェッチ要求を出しておらず、また命令到着待ち合わせ状態でもない場合

: `IC_fetch_L = 0`かつ`IC_miss_L = 0`

れる。すなわち、

`scope_next = 0`である。

(2) 分岐命令の発行

: `br_taken = 1`

この場合、第4図に示すように、アドレス`scope`は分岐命令を格納するキュー9のアドレスに設定される。したがって、`scope_next = branch_inst_entry`となる。

(3) この上述の状態以外の場合: `reset = 0`かつ`br_taken = 0`

この状態においては、アドレス`scope`は発行命令数に従ってシフトする。すなわち、

`scope_next = scope_L + issued_inst_count`

となる。ここで、「+」はモジュール12のモジュロ加算である。したがって、

`scope > 11 == 1`のとき、

`scope_next = scope_next - 12`

となる。

このスコープネクストロジック31の実現する論理は第7図に一覧にして示される。

ここでフラグ $scope > 11$, $queue_top > 11$ はアドレス $scope$ および $queue_top$ がキュー9を一巡りしたことを示す。

次に第8図を参照して命令フェッチロジック32の実現する論理について説明する。

(1) 命令フェッチ要求は、リセット状態のときまたは命令メモリが、命令フェッチ要求を受入れられる状態のとき発生される。すなわち、 $reset = 1$ かつ $IC_busy = 1$ のときに

$IC_fetch = 1$

となる。ここで「1」は否定を示す符号である。

(2) 命令フェッチ要求が出されないのは上述のごとくリセット状態または命令メモリが命令フェッチ要求を受入れない状態のときである。したがって上の状態(1)以外すべて

$IC_fetch = 0$

$dy = 0$

この状態では、次のサイクルは命令供給待機状態であり、次のサイクルで命令が与えられればその命令を書込む必要がある。フラグ IC_miss は1となる。

(4) 命令メモリから命令供給がされた場合
: $IC_ready = 1$

この状態は命令メモリから命令が供給されたことを示しており、次のサイクルにおいて命令待ち合わせ状態となる必要はなく、フラグ IC_miss は0となる。

(5) 動作状態において、分岐命令が発生せず、命令フェッチ要求も発生されず、また命令メモリからの命令供給指示も与えられない状態

: $br_taken = 0$ 、かつ $IC_fetch_L = 0$ 、かつ $IC_ready = 0$

この状態は、たとえば割込み処理などによる実行の中断状態などが生じており、フラグ IC_miss は前のサイクルの値を保持する必要があるため、

となる。次に命令ミスロジック33が実現する論理について第9図を参照して説明する。

信号 IC_miss は、命令メモリからの命令到着待ち状態であることを示すフラグである。この命令ミスロジック33は次のサイクルのフラグ IC_miss の状態を決定する。

(1) リセット状態: $reset = 1$

この状態においては、まだ何ら命令は発生されておらず命令を持つ必要もなく、フラグ IC_miss は0である。

(2) 動作時において、分岐命令が発生した場合

: $br_taken = 1$

この状態においては、次のサイクルで分岐先命令のフェッチが行なわれるため、命令待ち合わせ状態となることはなく、フラグ IC_miss は0となる。

(3) 命令フェッチ要求が出されても命令メモリから命令が供給されない場合

: $IC_fetch_L = 1$ かつ IC_rea

$IC_miss = IC_miss_L$ 、

となる。

次に第10A図および第10B図を参照してキューアペールロジック34が実現する論理動作について説明する。

(1) リセット状態: $reset = 1$

この状態においては、キュー9においては命令は何ら格納されていないため、キュー9には、アドレス $queue_top$ からの連続する4つのアドレス領域に4つの命令を書込むことができる。したがってこの場合、

$queue_avail = 1$

となる。

(2) 分岐命令が発生した場合: $br_taken = 1$

この状態においては、第4図に示すように、 $queue_top$ から始まるアドレスに分岐先命令を書込む必要があるため、空き領域が等価的に存在することになる。したがって、

$queue_avail = 1$

となる。

分岐が発生していない状態において、キューフルロジック（これは以下に説明する）がキュー9に空き領域（4エントリ分）が存在していることを示している場合には、

`queue_avail=1`となり、またこのキューフルロジックがキュー9が空き状態（4命令を書込むエントリ分の領域）にないことを示している場合には

`queue_avail=0`となる。すなわち、

この状態においては`queue_avail=!`
`queue_full`となる。

次に、キューフルロジックについて説明する。このキューフルロジックは第1図におけるキューアペールロジック34に含まれている。このキューフルロジックが実行する論理を第10B図を参照して説明する。

(1) このキューフルロジックはスコープネクストロジック31からのアドレス`scope`とキュートップロジック30からのアドレス`que`

となる。そうでない場合には、

`queue_full=TRUE`（1；空き領域無し）

となる。

(2) `scope_next > queue_top_in` :

この状態は第10D図に示す状態である。この状態では、`scope_next`と`queue_top_in`との差Cが4以上であればキュー9に命令を書込むことができる。したがって、

`scope_next - queue_top_in ≥ 4`ならば、

`queue_full=FALSE`

となる。

また、`scope_next - queue_top_in < 4`ならば、

`queue_full=TRUE`

となる。

(3) `scope_next = queue_top_in`

`ue_top_in`に従ってキュー9が空き領域を有しているか否かを判定する。ここで、スコープネクストロジック31から出力されるアドレス`scope`は次のサイクルにおける読出先頭アドレスを示しているため`scope_next`として説明する。

(1) 次のサイクルにおける読出先頭アドレス`scope_next`が次のサイクルにおける書込先頭アドレス`queue_top_in`よりも小さい場合

この場合、第10C図に示すように、命令が`queue_top_in`から書込まれるため、アドレス`scope_next`とキュー9の全エントリ数との和からアドレス`queue_top_in`との差が4以上あれば次のサイクルで命令を書込むことができる。したがって、

`scope_next + 12 - queue_top_in ≥ 4`

の場合には`queue_full=FALSE`（0；空き領域有）

この状態は次のサイクルにおける読出先頭アドレスと書込先頭アドレスとが等しい状態である。

このとき、後述のキューステイト（`queue_state`）に従って、キュー9の空き領域があるか否かの判別が行なわれる。すなわち、

`queue_state=NORMAL`ならば、

`queue_full=FALSE`

となる。また、`queue_state!=NORMAL`ならば、

`queue_full=TRUE`

となる。

次にキュー9の空き領域の有無を判定するための信号`queue_state`について説明する。

ここで、空き領域とは一度に命令メモリからフェッチされた複数の命令をすべて書込むことのできる領域であり、本実施例では最小4エントリの領域を示す。キュー9の状態は第1の状態の他に、第2の状態および第3の状態がある。まず、第1A図および第11B図を参照して第2の状態（`NORMAL`状態）について説明する。

第11A図および第11B図は、キュー9の第2の状態(NORMAL状態)のキュー9のアドレス`queue_top`および`scope`の位置関係を示す図である。第11A図において、アドレス`queue_top`は、キュー9のアドレス6を示し、アドレス`scope`はキュー9のアドレス2を示す。この状態は、

$scope < queue_top$
と表わせる。

第11B図において、命令が供給されず、命令の発行のみが行なわれた状態を示す。この場合、次のサイクルにおけるアドレス`scope_next`および`queue_top_in`は互いに等しくなる。この状態は、第2の状態(NORMAL状態)において生じたものであり、この状態も第2の状態と称される。この状態は、

$scope_next = queue_top_in$
と表わせられる。

したがって、キュー9の第2の状態(NORM

$scope_next = queue_top_in$

となる。この状態では、未発行の命令のみでキュー9が満たされている。この状態では、次のサイクルにおける命令書込を禁止する必要がある。この状態をも合わせて第3の状態(REVERSE状態)とよぶ。したがって、第3の状態(REVERSE状態)の条件は、

$scope \geq queue_top$
と表わせる。このキュー9の状態を信号`queue_state`によりモニタすることにより、前述のロジック`queue_full`の論理動作を通して、キュー9に空き領域があるか否かの判別を行なうことができる。

このキューステイトロジック50が実現する論理動作を第13図に一覧にして示す。以下、キューステイトロジック50の動作について説明する。

(1) リセット時の初期状態においては、アドレス`queue_top`および`scope`はキュー9のアドレス0に設定される。この状態は、

AL状態)にある条件は、

$scope \leq queue_top$

と表わせる。次に、第12A図および第12B図を参照して、キュー9の第3の状態(REVERSE状態)について説明する。

第12A図および第12B図はキュー9の第3の状態(REVERSE状態)におけるアドレス`scope`および`queue_top`の位置関係を示す図である。第12A図において、アドレス`scope`はキュー9のアドレス6を示し、アドレス`queue_top`は、キュー9のアドレス2を示す。この状態は、

$scope > queue_top$
と表わせる。

この状態において、命令発行が行なわれず、命令供給が行なわれた状態を考える。すなわち、第12A図においてキュー9のアドレス2から命令が供給された場合を考える。この場合、第12B図に示すように、次のサイクルにおけるアドレスは、

キュー9の第1の状態であるが、キュー9へ命令を書込むことができるため、信号`queue_state`は第2の状態に設定される。すなわち、
 $reset = 1$ ならば、
 $queue_state = NORMAL$
となる。

(2) アドレス`queue_top`が、キュー9のアドレス領域を一巡し、かつアドレス`scope`がまだキュー9のアドレス領域を一巡しない場合:

このとき、第13図の(2)に示すように、キュー9は第3の状態(REVERSE状態)となる。したがって、

$queue_top > 11 = 1$ 、かつ
 $scope > 11 = 0$ のとき、
 $queue_state = REVERSE$
となる。

(3) アドレス`scope`がキュー9のアドレス領域を一巡した場合:

この状態では、アドレス`scope_next`

がアドレス `queue_top_in` を超えることはないで、キュー9は第2の状態(NORMAL状態)となる。したがって、

```
scope > 11 == 1 ならば、
queue_state = NORMAL
となる。
```

(4) 分岐が発生した場合：

この状態においては、分岐命令から始まるキュー9のアドレス領域に次の分岐先命令を格納する必要がある。したがってこの状態ではアドレス `scope_next` および `queue_top_in` は等しくされるが、命令の書込を可能とするために、キュー9の状態は第2の状態(NORMAL状態)となる。したがって、

```
br_taken = 1 ならば、
queue_state = NORMAL
となる。次に、キュー9の「有効性フラグ」部分に書込まれるフラグを決定する命令アベールロジック35が実現する論理について第14図を参照して説明する。
```

令の供給の有無に対応する。したがって、

```
IC_fetch_L = 1、ならば、
inst_avail_in = IC_read
y
となる。
```

(4) 命令到着待ち状態の場合：

この状態は、そのサイクルでの命令フェッチ要求の有無にかかわらず生じる。このときも、アドレス `queue_top` から始まる命令の有効/無効は命令メモリからの命令供給の有無に対応する。したがって、

```
IC_miss_L = 1、ならば、
inst_avail_in = IC_read
y
となる。
```

(5) 上述の4状態のいずれの状態でもない場合：

この状態においては、アドレス `queue_top` から始まる4エントリに書込まれる命令は要求する命令ではないため、有効性フラグはオフと

第14図は命令アベールロジック35が実現する論理を一覧にして示す図である。

(1) 初期状態：`reset = 1`

この状態では、まだキュー9に有効な命令は何ら格納されていないため、有効性フラグはオフ(0)にされる。したがって、

```
reset = 1 ならば、
inst_avail_in = 0
となる。
```

(2) 分岐が発生した場合、

この状態では、分岐命令が格納されたアドレスから続く4エントリの命令の発行を禁止する必要がある。したがってこの場合には有効性フラグはオフ(0)とする。したがって、

```
br_taken = 1、ならば、
inst_avail_in = 0
となる。
```

(3) 命令フェッチ要求が出された場合：

この状態では、アドレス `queue_top` から始まる命令の有効/無効は命令メモリからの命

される。したがって、

```
else, inst_avail_in = 0
となる。
```

第15図はキュー初期化ロジック36の実現する論理を一覧にして示す図である。以下、第15図を参照してキュー初期化ロジック36の論理動作について説明する。ここで、キューの初期状態(第1の状態)とは、キューのリセット状態、分岐命令が発生した状態、および命令待ち状態のいずれかの状態を示す。すなわち、キュー9に必要とされる命令が書込まれていない状態を狭わしている。

(1) リセット状態：

この状態においては、まだキュー9には何ら命令は書込まれていないため、キュー9は初期状態にある。したがって、

```
reset = 1、ならば
queue_init_state = 1
となる。
```

(2) 分岐が発生した場合：

この状態においては、キュー9は分岐先命令を格納する必要があるが、この命令は、まだキュー9に書込まれていないため、キュー9は初期状態にある。したがって、

`br_taken=1`、ならば、
`queue_init_state=1`、
 となる。

(3) 命令フェッチ要求が出されても命令メモリから命令が供給されない場合：

この状態においてはキュー9は命令待ち状態となるため、キュー9は初期状態となる。したがって、

`IC_fetch_L=1`、かつ
`IC_ready=0`、ならば、
`queue_init_state=1`、
 となる。

(4) 命令フェッチ要求が出されかつ命令メモリから命令が供給された場合：

この状態においてはキュー9には所望の必要とされる命令が書込まれたことにより、キュー9は

`IC_ready=1`ならば、
`queue_init_state=0`
 となる。

(7) リセット状態になく、分岐発生状態でもなく、命令フェッチ要求を行なっておらず、また命令待ち合わせ状態でもない場合：

この状態においては、キュー9は命令メモリからの命令の供給の有無にかかわらず、キュー9は次のサイクルの状態では、そのときの状態を保持すべき状態にあり、またすなわち、第6図に示すようにアドレス`queue_top`は同一アドレスを繰り返して示す必要がある。したがって、

`reset=0`、かつ`br_taken=0`、
 かつ
`IC_fetch_L=0`かつ`IC_miss_L=0`、ならば、
`queue_init_state=queue_init_state_L`
 となる。

なおロジック30ないし36および50が実現

初期状態ではない。したがって、

`IC_fetch_L=1`、かつ
`IC_ready=1`、ならば、
`queue_init_state=0`、
 となる。

(5) キュー9が命令待ち状態にあり、かつ命令メモリから命令が供給されない場合：

この状態では、キュー9はフェッチ要求した命令の到着を待っている。したがって、キュー9は初期状態にある。したがって、

`IC_miss_L=1`、かつ
`IC_ready=0`、ならば
`queue_init_state=1`
 となる。

(6) キュー9が命令待ち状態にありかつ命令メモリが命令を供給した場合：

この状態では、キュー9にはフェッチ要求した命令が書込まれたため、キュー9は初期状態ではなくなる。したがって、

`IC_miss_L=1`、かつ

する論理を一箇にして示しているが、これらの論理を実現するための具体的構成は当業者であれば容易に作成することができるであろう。

命令デコード26は、Lラッチ回路14～16から与えられるアドレス`ID_PC_L`、命令`ID_IR_L`および有効性フラグ`ID_inst_avail_L`をデコードし、並列処理可能な命令を検出し、該命令を対応の機能ユニットへ転送するとともに、信号`br_taken`、データ`issued_inst_count`および`branch_inst_entry`を発生する。

この第1図に示す回路構成は1サイクルですべて所望の動作を実行する。分岐発生時において、分岐命令と同時に命令デコード26に与えられた命令のうち分岐命令以後の命令については、デコード動作を行なっても機能ユニットへの発行を停止する構成であってもよく、また機能ユニットへその命令を与える一方そのサイクルにおける機能ユニットの動作を禁止する構成であってもよい。

次に、各ロジック30ないし36および50の動

作について図面を参照して説明する。

第16図は第3図に示すキュー、`queue_top`および`scope`の基本的な動きに対応する各ロジックの動作を示す信号波形図である。第16図には、各サイクルと第3図に示す各ステップとの対応関係を合わせて示す。また、各サイクルの初めはクロック信号Tにより決定される。この信号波形図においてサイクルの途中でロジックの出力の変化は、クロック信号Lに反応して生じている。

(1) サイクル0および1:

このサイクルはリセット信号`reset`が“1”にあり、キュー9は初期状態にある。したがって、信号`queue_init_state`がオン

(1)、アドレス`queue_top`および`scope`は0であり、また有効性フラグ`inst_avail_in`は0である。

また、リセット状態においては、命令メモリに対するフェッチ要求は行なわれないため、命令フェッチ要求フラグ`IC_fetch`はオフ(0)

`inst_avail_in`がオン状態となる。このとき、キュー初期状態フラグ`queue_init_state`はオン状態にあり、キュー9が初期状態にあることを示している。

(4) サイクル4:

信号`IC_fetch`および`IC_ready`がともにオン状態となったことにより、キュー9の初期状態が解除され、信号`queue_init_state`がオフ(0)とされ、キュー9に対する命令等の書込が実行される。すなわち、アドレス`queue_top`が示すアドレス0からの領域に命令メモリから供給された命令`IC_data`および命令フェッチステージから伝達された各命令の論理アドレス`IF_PC_L`および命令アペールロジック35からの有効性フラグ`inst_avail_in`が書込まれる。

またこのときアドレス`scope`から始まるアドレス0に書込まれた命令およびアドレスおよび有効性フラグが読出され、ラッチ回路14ないし16へ与えられる。このラッチ回路14ないし1

である。また、キュー9はリセット状態においては空き領域が十分に存在するため、信号`queue_avail`は空き領域があることを示すため“1”である。

(2) サイクル2:

このサイクル2においてリセット信号`reset`が“0”となり、リセットが解除される。このリセットの解除はサイクルの途中で行なわれており、各ロジックの出力状態はサイクル1と同様である。

(3) サイクル3:

このサイクルにおいては、命令フェッチロジック32からの命令フェッチ要求`IC_fetch`をオン状態とする。これにより命令メモリに対する命令フェッチ要求が行なわれる。この発行された命令フェッチ要求に対して命令メモリから命令が供給される。命令メモリからは命令供給を示す信号`IC_ready`がオン状態となる。このオン状態の`IC_ready`に反応して、命令アペールロジック35からの出力される有効性フラグ

6へ与えられたデータはクロック信号Lに反応して命令デコード26へ与えられてそこで解説される。その結果、サイクル4において2つの命令が並列処理可能であるとして対応の機能ブロックに発行される。これにより、命令デコード26からは発行命令数が2であることを示す信号`issued_inst_count`が発生され、スコープネクストロジック31へ与えられる。アドレス`scope_next`は2となる。また、アドレス`queue_top`は4を加算され、4となる。

(5) サイクル5:

このサイクルにおいては、アドレス`queue_top`はアドレス4を示しており、またアドレス`scope`はアドレス2を示している。サイクル4と同様に命令メモリから命令が供給され、アドレス4ないし7に命令が書込まれる。このとき同時に、キュー9からアドレス2からの命令が読出され、ラッチ回路14ないし16を介して命令デコード26へ与えられる。命令デコード26からは1つの命令が機能ユニットへ発行される。し

たがって、スコープネクストロジック31から出力されるアドレス`scope`は3となる。また、キュートップロジック30から出力されるアドレス`queue_top_in`は8となる。

(6) サイクル6:

このサイクル6においてもサイクル4と同様に命令メモリから命令が供給される。これにより、キュー9のアドレス8ないし11に命令が書込まれる。

一方、命令デコーダ26からは機能ユニットへ発行される命令が存在しなかったため、発行命令数0を示す信号`issued_inst_count`が発生されスコープネクストロジック31へ与えられる。

キュートップロジック30からの次のサイクルのアドレス`queue_top_in`はアドレス0を示す。スコープネクストロジック31から出力されるアドレス`scope_next`は3である。このとき、キュー9のアドレス3には未発行の命令が残る。したがって、キューアペールロジック

ック34は、そこに含まれるキューフルロジックの機能によりキュー9に空き領域がないことを示すために、信号`queue_avail`をオフ(0)に設定する。これにより、次のサイクルにおけるキュー9への命令の書込が禁止される。

(7) サイクル7:

このサイクルにおいては、キュー9のアドレス3から命令が読出され、命令デコーダ26から3つの命令が発行される。これにより、スコープネクストロジック31から出力されるアドレス`scope_next`は6となる。アドレス`queue_top_in`は、そのとき信号`queue_avail`がオフ状態であったため変更せず0である。このとき、アドレス`scope_next`が6であり、一方アドレス`queue_top_in`が0であり、キュー9に空き領域が生じたため、キューアペールロジック34からの出力信号`queue_avail`はオンとなり、次のサイクルにおけるキュー9への命令の書込が許可される。

(8) サイクル8:

アドレス`queue_top`はサイクル7と同様0であるが、キュー9は命令を書込むことができるため(`queue_avail`はオン状態)、キュー9のアドレス0ないし3の領域に命令が書込まれる。このとき、アドレス`scope`は6であり、キュー9はアドレス6を示しており、このキュー9のアドレス6からの命令が命令デコーダ26へ与えられる。

なお、このとき、命令フェッチ要求`IC_fetch`が連続して発行されるため、サイクル7とサイクル8とで命令メモリから与えられる命令の内容が異なることも考えられる。これは、キューアペールロジック34からの信号`queue_avail`を命令フェッチステージへ与え、この信号`queue_avail`がオフの場合には、命令フェッチステージに含まれるプログラムカウンタ(`IF_PC`)のカウンタ動作を禁止し、サイクル7とサイクル8とで命令メモリの同一の内容が与えられるように構成すれば、同一命令を命令

メモリから与えることができる。

なお、各ロジック30ないし36および50は与えられた信号を論理処理しているだけであり、クロック信号TまたはLとの同期動作は行なっていない。

第17図は、第4図に示す分岐発生時における各ロジックの動作を示す信号波形図である。第17図において第4図の各ステップが各サイクルに対応づけて示される。以下、第17図を参照して分岐発生時におけるキュー、`queue_top`および`scope`の動きについて説明する。

(1) サイクル0:

アドレス`queue_top`は10であり、アドレス`scope`は2である。したがってキュー9からは命令8、9、10および11がアドレス10、11、0および1に書込まれる。一方、キュー9のアドレス2ないし5の命令0ないし3が読出される。このサイクル0において、キュー9のアドレス4に格納されていた分岐命令23により分岐が発生した場合を想定する。この場合、信

号 `br_taken` がオン (1) となる。この分岐が生じたことにより、命令アペールロジック 35 からの有効フラグ `inst_avail_in` はオフ (0) となる。

また、この分岐発生により、キュートップロジック 30 から出力されるアドレス `queue_top_in` は分岐命令 2 が格納されていたアドレス `branch_inst_entry` に従って 4 となり、また同様に、スコープネクストロジック 31 から出力されるアドレス `scope_next` も 4 となる。

またキュー初期化状態ロジック 36 の出力 `queue_init_state` もこの分岐発生により初期状態 (第 1 の状態) を示すために 1 となる。また、キューアペールロジック 34 の出力へはこの分岐発生によりキューステイトロジック 50 から `NORMAL` 状態を示す信号が与えられ、信号 `queue_avail` は 1 のままである。

(2) サイクル 1 :

このサイクルにおいては、アドレス `queue`

クルにおいては分岐は生じないため、命令デコード 26 からの信号 `br_taken` はオフとなる。これにより、命令アペールロジック 35 からの有効性フラグ `inst_avail_in` はオンとなる。

(3) サイクル 2 :

このサイクルにおいては、分岐先命令が供給され、キュー 9 に格納されるとともに、関連の有効性フラグ `inst_avail_in` もオンとなる。このときアドレス 4 から始まる命令すなわち分岐先命令が命令デコード 26 へ与えられデコードされる。このサイクルにおいては、信号 `queue_init_state` はオフとされているため、その処理内容に応じてアドレス `queue_top` および `scope` はそれぞれ変更される。ここで、サイクル 2 においては、信号 `IC_ready` がオンのため、サイクル 2 において分岐先命令が供給されていることを示している。

第 18 図は第 5 図に示す動作状態における各ロジックの動作を示す信号波形図である。以下、第

`_top` および `scope` はともに 4 に設定され、またキュー 9 のアドレス 4 ないし 7 の有効性フラグに 0 が書込まれる。この分岐が発生した場合には、分岐先命令のアドレスがこのサイクルで命令フェッチステージから命令メモリへ与えられ、分岐先命令はその次のサイクルに命令メモリから供給される。したがって、この分岐先命令はこのサイクルではキュー 9 には書込まれない。

このサイクル 1 においては、アドレス 4 から始まる命令は命令デコード 26 へ与えられるが、この命令内容は機能ユニットへ発行／実行してはならないものである。このため、命令デコード 26 は、関連の有効性フラグ `inst_avail` がオフ状態となるためその発行を行なわない。

さらに、キュー初期化ロジック 36 からの出力信号 `queue_init_state` はオン

(1) であるため、キュートップロジック 30 およびスコープネクストロジック 31 から出力されるアドレス `queue_top_in` および `scope_next` は 4 を保持する。またこのサイ

クル 18 図を参照して、発生された命令フェッチ要求に対し命令メモリから命令が供給されない場合の動作について説明する。なお、第 18 図において第 5 図に示すステップも合わせて各サイクルに対応づけて示されている。

(1) サイクル 0 :

このサイクルにおいてアドレス `queue_top` が 0、アドレス `scope` が 6 である。このサイクル 0 において発生された命令フェッチ要求 (`IC_fetch` がオン) に対し、命令メモリから命令が供給されない場合を想定する。このとき、命令メモリからの信号 `IC_ready` はオフとなり、また命令メモリへのアクセスを禁止するため、信号 `IC_busy` はオンとなる。このオフ状態の信号 `IC_ready` に応答して有効性フラグ `inst_avail_in` がオンからオフ状態へ変わる。

(2) サイクル 1 :

このサイクルにおいては、サイクル 0 において、`IC_fetch` が 1 でありかつ `IC_read`

yが0のため、信号IC_missが1となり、キュー9が命令待ち状態にあることを示す。またキュー初期化ロジック36からの出力queue_init_stateは、サイクル0において、命令要求が発生されても命令供給がされなかったことに応答して“1”となりキュー9が初期状態（第1の状態であり命令待ち状態）に設定され、アドレスqueue_topの変更が禁止される。このサイクルにおいては、アドレス4から始まるキュー9の領域にはオフ（0）の有効性フラグinst_avail_inが書込まれ、該領域に格納された命令の発行／実行が禁止される。

一方、アドレスscopeが示すアドレスキューからの命令は命令デコード26へ与えられ、2つの命令が発行され、次のサイクルにおけるアドレスscope_nextは11に変更される。

(3) サイクル2：

この状態においても待ち状態であり、IC_fetchはオフ、信号IC_missはオン、および信号queue_init_stateはオ

ン状態である。この状態においては、アドレス4から始まる4エントリ領域に命令無効を示すフラグinst_avail_inが書込まれる。

一方、アドレスscopeから始まる命令は読出され、命令デコード26へ与えられ、2つの命令が発行され、次のサイクルにおける書込先頭アドレスscope_nextは1に変更される。

このサイクル2において、命令メモリから命令が供給されたとする。このとき、信号IC_readyがオン状態となり、信号IC_busyはオフ状態となる。これに responding、有効性フラグinst_avail_inはオン状態へ移行する。この状態により、キュー9は命令待ち状態が解除される。

(4) サイクル3：このサイクルにおいては、命令メモリから到達した命令がアドレスqueue_topが示す4から始まる領域に登録され、またその対応の有効性フラグinst_avail_inも1に設定される。このとき、アドレスscopeが示すアドレス位置からの命令に対す

る命令デコード26による機能ユニットへの発行が行なわれる。このサイクル3においてはキュー9は初期状態から解除されたため、初期化状態信号queue_init_stateはオフとされ、アドレスqueue_topの変更が再開される。また、信号IC_fetchもオン状態とされ、命令フェッチ要求も再開される。

上述のように、複数の命令をそのアドレスおよび有効性フラグとともに同時に書込かつ読出を行なうことのできるキューを設け、このキューの書込／読出ポイントを、分岐発生、命令供給状態およびリセット状態の第1の状態のいずれにあるかおよびキューに空き領域があるか否かに従ってポイントの制御および命令の機能ユニットへの発行可否を制御するように構成したため、命令デコードへ効率的に命令を供給することができ、高速で命令を実行することのできるスーパー scalerを得ることができる。次に、この複数の命令を同時に書込／読出することのできるキュー9の構成および動作について説明する。

第19図は命令メモリ1からフェッチされる命令とキュー9へ書込まれる命令との対応関係を概念的に示す図である。まず、第19図を参照して、キュー9へのアドレスIF_PC_L、命令IC_dataおよび有効性フラグinst_avail_inの書込動作について説明する。

命令フェッチステージ2からは命令フェッチ要求IC_fetchに responding、命令メモリ1のアドレスIF_PCが発生されて命令メモリ1へ与えられる。命令メモリ1からはこのアドレスIF_PCに従って4つの命令IC_1、IC_2、IC_3およびIC_4が同時に読出される。この4つの命令IC_1～IC_4は並列に命令IC_dataとしてキュー9の命令領域へ与えられる。

一方、命令フェッチステージ2からはこのアドレスIF_PC_Lがキュー9へ供給される。このアドレスIF_PC_Lは4つの命令IC_1～IC_4の各アドレスを示しており、これらのアドレスが並列にキュー9のアドレス領域へ与え

られる。この命令は任意の長さであってもよいが、キュー9のエントリの効率的利用からは命令メモリ1からの命令の長さはすべて同一の長さに設定される。

命令メモリ1から発生される信号IC_{bus}yは第1図に示す命令フェッチロジック32へ与えられる。命令メモリ1から発生される信号IC_{ready}は命令アベールロジック35へ与えられる。この命令アベールロジック35へ与えられる他のロジックからの信号は図示していない。

キュー9は、これらのアドレスIF_{PC}L、命令IC_{data}および有効性フラグinst_{avail}inを受け、書込イネーブル信号wen（これは第1図のキューアベールロジック34から発生される）に応答して、アドレスqueue_{top}が示すアドレスから始まる4つのエントリ領域へ同時にこれらを書込む。

このときまた、アドレスscopeから始まる4つのエントリの内容が読出される。このアドレスIF_{PC}Lは、クロック信号Lに応答して

確定状態になるものではあるが、クロック信号Tに反応して発生されるものであってもよい。

キュー9に書込まれた命令およびアドレスが同時にまた読出される場合もあり、このキュー9から読出された命令およびアドレスは、クロック信号Lに応答して保持動作を実行するラッチ回路14および15で保持されるため、このキュー9の書込データはラッチ回路14および15の保持動作前に確定していれば良い。したがって、このアドレスIF_{PC}Lが同時にまた読出されるときにはこのラッチ回路のラッチタイミングを少し遅らせておけば、アドレスIF_{PC}Lがクロック信号Lに応答して確定状態とされるものであっても何ら誤動作は生じない。

第20図はキュー9の命令領域へ与えられる命令IC_{data}の配置を示す図である。第20図を参照して、同時に読出される4つの命令IC₁～IC₄の各々は、各ビットごとに集められる。すなわち、命令IC_{data}の第0ビット領域には4つの命令IC₁～IC₄の第0

ビットIC₁₀～IC₄₀が順次格納され、以下この順番で第32ビットまで同様に配置される。ここで、命令IC₁～IC₄はすべて32ビット構成の場合が一例として示されている。

第21図はキュー9のアドレス領域へ伝達される命令アドレスIF_{PC}Lの配置を示す図であり、命令IC_{data}と同様各命令ごとにビットが順番に配置される。

第22図は有効性フラグの配置を示す図である。この有効性フラグinst_{avail}inは4エントリに対応して4ビット構成とされ、すべて同一の値をとる。

第23図はキューの全体の構成を概略的に示す図である。第23図においてキュー9は、アドレスqueue_{top}をデコードし、4エントリを選択するための信号を発生する書込デコーダWRDと、アドレスscopeをデコードし、命令を読出すべき4エントリを選択する信号を発生する読出デコーダRDDと、各々が1エントリ領域を構成し、アドレス、命令および有効性フラグを

格納する並列に設けられたエントリ記憶装置92～92-11を含む。

書込デコーダWRDおよび読出デコーダRDDは、それぞれ与えられたアドレスqueue_{top}およびscopeに応答してその変化に従ってデコード動作を実行する。すなわちこれらの書込デコーダWRDおよびRDDは非同期デコード動作をスタティックに行なっており、特にこれらの書込デコードおよびWRDおよび読出デコーダRDDの動作タイミングを規定する制御信号は用いられてはいない。

書込デコーダWRDには12本の書込ワード線WD₀～WD₁₁が接続され、アドレスqueue_{top}に従ってこれらの12本の書込ワード線WD₀～WD₁₁のうちの1本を選択状態とされる。読出デコーダRDDも12本の読出ワード線RD₀～RD₁₁を有しており、アドレスscopeに応答してこれらの12本の読出ワード線RD₀～RD₁₁のうちの1本を選択状態とする。後に詳細に説明するが、1つの書込ワード線WD

i (i = 0 ~ 11) および1本の読出ワード線 RD i にはそれぞれ連続して隣接する4つのエントリ記憶装置が接続される。したがって、1本の書込ワード線または読出ワード線が選択状態とされることにより、同時に4つのエントリ記憶装置が選択状態とされる。

エントリ記憶装置 92-0 ~ 92-11 の各々は、データ書込ポートを選択するための書込アクセスゲート w 0、w 1、w 2、および w 3 と、データ読出経路を選択するための読出アクセスゲート r 0、r 1、r 2 および r 3 を含む。この書込アクセスゲート w j (j = 0 ~ 3) および読出アクセスゲート r 0 ~ r 3 のいずれか1つを選択することによりデータの書込/読出ポートが選択される。4つの隣接するエントリ記憶装置の書込アクセスゲートの異なる書込アクセスゲートが1本の書込ワード線 WD i に接続される。同様に、4つの隣接するエントリ記憶装置の異なる読出アクセスゲートが1本の読出ワード線 RD i に接続される。

記憶装置 92-0 ~ 92-11 の書込命令入出力ポート i w i および i w o はすべて共通の命令書込ビット線 IWB に接続され、読出命令入出力ポート i r o および i r i は共通に命令読出ビット線 IRB に接続される。

記憶装置 92-0 ~ 92-11 の書込アドレス入出力ポート p w i および p w o はアドレス書込ビット線 PWB に接続され、記憶装置 92-0 ~ 92-11 の読出アドレス入出力ポート p r o および p r i はアドレス読出ビット線 PRB に接続される。

記憶装置 92-0 ~ 92-11 の書込有効性フラグ入出力ポート a w i および a w o は有効性フラグ書込ビット線 AWB に接続され、読出有効性フラグ入出力ポート a r i および a r o は有効性フラグ読出ビット線 ARB に接続される。

命令書込ビット線 IWB、命令読出ビット線 IRB、アドレス書込ビット線 PWB、およびアドレス読出ビット線 PRB は4命令分のデータを伝達することのできるビット幅を有しており、本実

たとえば、読出ワード線 WD 0 には記憶装置 92-0 の書込アクセスゲート w 0、記憶装置 92-1 の書込アクセスゲート w 1、記憶装置 92-2 の書込アクセスゲート w 2、および記憶装置 92-3 の書込アクセスゲート w 3 が接続される。また、読出ワード線 RD 0 には、記憶装置 92-0 の読出アクセスゲート r 0、記憶装置 92-1 の読出アクセスゲート r 1、記憶装置 92-2 の読出アクセスゲート r 2 および記憶装置 92-3 の読出アクセスゲート r 3 が接続される。

エントリ記憶装置 92-0 ~ 92-11 の各々は、書込データ入力ポート i w i、書込データ出力ポート i w o、読出データ入力ポート i r i、読出データ出力ポート i r o、書込アドレス入力ポート p w i、書込アドレス出力ポート p w o、読出アドレス出力ポート p r o、読出アドレス入力ポート p r i、書込有効性フラグ入力ポート a w i、書込有効性フラグ出力ポート a w o、読出有効性フラグ出力ポート a r o、および読出有効性フラグ入力ポート a r i を含む。

施例においては、 $4 \cdot 32 = 128$ のビット幅を有している。有効性フラグ書込ビット線 AWB および有効性フラグ読出ビット線 ARB は4ビットの幅を有している。これらの読出ビット線および書込ビット線上に4命令分の内容が同時に伝達される。

記憶装置 92-0 ~ 92-11 の各々の入力および出力ポートはともに4つのポートを有しており、この4つのポートのうちの1つが書込アクセスゲート w 0 ~ w 3 および読出アクセスゲート r 0 ~ r 3 より選択される。したがって、同時に選択状態とされる4つの記憶装置は互いに異なる入力ポートおよび出力ポートが選択状態とされる。

情報の入出力を行なうために、第1図に示すキューアペールロジック34から発生される書込イネーブル信号 wen に応答して所定のタイミングで書込パルス WP を発生する書込パルス発生器 WPG と、書込パルス発生器 WPG からの書込パルス WB に応答してオン状態となる、トライステートバッファ BF 1、BF 2 および BF 3 が設けら

れる。このトライステートバッファBF1～BF3は書込パルスWPが発生されない場合その出力状態をハイインピーダンス状態とする。

トライステートバッファBF1は、命令メモリからの命令IC_dataを受け、命令書込ビット線IWB上へ伝達する。トライステートバッファBF2は、アドレスIF_PC_Lをアドレス書込ビット線PWB上へ伝達する。トライステートバッファBF3は有効性フラグinst_avail_inを有効性フラグ書込ビット線AWB上へ伝達する。命令読出ビット線IRBからは命令instが出力され、ラッチ回路14へ伝達される。読出ビット線PRBからはアドレスPCが読出され、ラッチ回路15へ伝達される。読出ビット線ARB上の有効性フラグinst_availはラッチ回路16へ接続される。

次にこの第23図に示すキューの動作について簡単に説明する。

まずデータ書込動作について説明する。アドレスqueue_topに従って、書込デコードW

RDにより1本の書込ワード線WDiが選択状態とされる。これにより、連続して隣接する4つのエントリ記憶装置が選択されるとともにそれぞれの異なる入力ポートが選択状態とされる。これにより、4つのエントリ記憶装置へ同時に書込を行なうことが可能となる。次いで、キュー9に空領域が存在するとキューアペールロジック34からの書込イネーブル信号wenがオン状態になり、書込パルス発生器WPGから所定のタイミングで書込パルスWPが発生され、トライステートバッファBF1ないしBF3が導通状態とされ書込データが選択されたエントリ記憶装置の対応のポートへ伝達され、4エントリに命令、アドレスおよび有効性フラグが同時に書込まれる。

読出動作も同様であり、アドレスscopeが与えられると読出デコードRDDは1本の読出ワード線RDiを選択し、4つのエントリ記憶装置の異なる読出アクセスゲートを選択する。これにより4つのエントリ記憶装置から同時に4つのエントリ内容を読出すことができる。この読出され

たデータは読出ビット線IRB、PRBおよびARBを介して対応のラッチ回路14、15および16へ伝達される。

第24図は1個のエントリ記憶装置の構成を示す図である。第24図において1個のエントリ記憶装置92を総称的に示す。命令記憶装置93-1、アドレス記憶装置93-2および有効性フラグ記憶装置94を含む。命令記憶装置93-1およびアドレス記憶装置93-2は同一の構成を有し、書込ポート選択ゲートwbと、読出ポート選択ゲートrbと、書込アクセスゲートw0～w3および読出アクセスゲートr0～r3を含む。ゲートwb、rbの<3:0>は入出力ポートが、ともに0ないし3の4つあることを示しており、また<31:0>は32ビットのデータが、1つの選択されたポートから出力されることを示している。したがって、命令書込ビット線IWBおよび読出ビット線IRB、アドレス書込ビット線PWBおよびアドレス読出ビット線PRBの128ビットの信号線のうち32ビットの信号線が1つ

の記憶装置により使用される。どの32ビットの信号線が使用されるかはアクセスゲートwb0～wb3およびr0～r3の選択により決定される。

有効性フラグ記憶装置94は書込ポート選択ゲートwbおよび読出ポート選択ゲートrbと、書込アクセスゲートw0～w3および読出アクセスゲートr0～r3を含む。この書込ポート選択ゲートwbは、ポートを0ないし3の4つ有しており、1つの選択ポートから1ビットの有効性フラグが入出力される。すなわち、アクセスゲートw0～w3およびr0～r3を選択することにより、フラグ書込ビット線およびフラグ読出ビット線AWBおよびARBの4ビットのうち1ビットが、1つの有効性フラグ記憶装置94により使用される。

第25図は命令記憶装置93-1およびアドレス記憶装置93-2の構成をより詳細に示す図である。この記憶装置93-1および93-2は同一の構成を有しており、記憶装置93を総称的に示す。第25図を参照して、命令記憶装置93-

1およびアドレス記憶装置93-2はともに、32ビットのビット記憶装置95-0~95-31を含む。このビット記憶装置95-0~95-31の各々は、命令またはアドレスの第0ビットないし第31ビットをそれぞれ記憶する。ビット記憶装置95-0~95-31の各々は、書込ポート選択ゲートwb0、wb1、wb2およびwb3と、読出ポート選択ゲートrb0、rb1、rb2およびrb3と、書込アクセスゲートw0~w3および読出アクセスゲートr0~r3を含む。このゲートwb0~wb3およびrb0~rb3の選択がアクセスゲートw0~w3およびr0~r3の選択により決定される。このビット記憶装置95-0~95-31の書込アクセスゲートw0~w3にはそれぞれ同一のアクセスゲート選択信号w0~w3（信号線とその上に伝達される信号とを同一の符号で示す）が伝達され、また読出アクセスゲートr0~r3には同一の読出ゲート選択信号r0~r3が伝達される。これにより、命令またはアドレスの32ビットが同時に選

択される。

有効性フラグ記憶装置94はこのビット記憶装置95-0~95-31と同一の構成を有している。

第26図はビット記憶装置および有効性フラグ記憶装置の具体的構成の一例を示す図である。第26図において、ビット記憶装置（および有効性フラグ記憶装置）は、書込ポート選択トランジスタGT1~GT4と、読出ポート選択トランジスタRT1~RT4と、1ビットのデータを記憶する記憶素子MEを含む。記憶素子MEは、反並行または交差結合された2つのインバータ1V1および1V2を含む。すなわち記憶素子MEはインバータラッチにより構成される。

ゲートトランジスタGT1はポート選択信号w0をそのゲートに受け、メモリMEをポートwb0に接続する。ゲートトランジスタGT2は、ポート選択信号w1に反応して記憶素子MEをポートwb1に接続する。ゲートトランジスタGT3は、ポート選択信号w2に反応し記憶素子MEを

ポートwb2に接続する。ゲートトランジスタGT4はポート選択信号w3に反応して記憶素子MEをポートwb3に接続する。

読出経路も同様であり、トランジスタRT1~RT4はそれぞれ読出ポート選択信号r0~r3に反応して記憶素子MEをポートrb0~rb3へそれぞれ接続する。

このビット記憶装置および有効性フラグ記憶装置の異なるエントリ間の接続形態を第27図に示す。第27図に示すように、このポートwb0~wb3およびrb0~rb3はそれぞれ同一ビットの記憶装置に対して共通に設けられており、また有効性フラグ記憶装置においてはすべての記憶装置に対して共通に設けられる。

ビット記憶装置（有効性フラグ記憶装置）の動作について説明する。書込ワード選択信号WRDにより1本の書込ワード線が選択された場合、この第26図の構成において1個のゲートトランジスタGT1がオン状態となり、記憶素子MEがポートwb0~wb3のいずれかに接続される。こ

れにより、選択されたポートを介して記憶素子MEへのデータの書込が行なわれる。データ読出も同様であり、読出デコードRDDにより1個の読出ワード線が選択され、応じてトランジスタRT1~RT4のいずれかがオン状態となり、記憶素子MEがポートrb0~rb3のいずれかに接続される。これによりデータの読出が行なわれる。

第27図に示すように、1本の書込ワード線WDiを選択状態とすることにより、隣接するエントリ記憶装置においては異なるポートが選択される。たとえば第27図において記憶素子ME1は選択信号WDiによりポートwb0に接続され、記憶装置ME2がポートwb1に接続される。これにより、4つの異なる命令の各ビットを4つの異なる記憶素子へ同時に書込むことができる。データ読出も同様である。1本の読出ワード線RDiを選択状態とすることにより、記憶素子ME1、ME2、…が異なるポートへ接続され、同時に4つのエントリの内容がそれぞれ異なるポートに伝達されるため、同時に4つの命令、アドレスおよ

び有効性フラグを読出すことができる。

次に、このキューにおけるデータ読出時の動作について第28図を参照して説明する。

この第28図に示す動作波形図においては、命令読出ビット線IRB、アドレス読出ビット線PRBおよび有効性フラグ読出ビット線ARBが読出ビット線RBとして代表的に示されている。アドレスscopeが与えられると、読出デコードRDDの出力は、このアドレスscopeの変化状態に従って変化し、ある時間が経過するとその出力状態が確定し、1本の読出ワード線RDが選択状態となる。この第28図に示す動作波形図においては、すべての読出ワード線の信号波形図がすべて示されており、選択状態および非選択状態の読出ワード線の信号状態が示されている。この選択ワード線（たとえばRD0）上の信号電位が確定すると、エントリ記憶装置の読出アクセスゲートが選択状態とされる。この場合、記憶装置92-0～記憶装置92-3のアクセスゲートr0、r1、r2およびr3がそれぞれ選択され、それ

ぞれの異なるデータ読出ポートが選択状態とされ、対応のポート上にデータが読出される。この後、トランジスタRT1～RT4により読出ビット線RB上の信号状態が確定する。この読出ビット線RB上の信号電位はLラッチ回路14～16へ与えられる。

第29図はデータ書込時の動作を示す信号波形図である。次にデータ書込動作について第29図を参照して説明する。

アドレスqueue_topが与えられると、書込デコードWRDは、このqueue_topに従ってデコード動作を開始し、ある時間が経過した後、書込ワード線WD上の信号電位が確定状態となる。これにより、データ書込を受けるべき4つのエントリが選択され、かつこの4つのエントリ記憶装置のそれぞれの異なるポートがアクセスゲートを介して選択状態とされる。この書込ワード線WDの信号電位が確定した後、キュー9に空領域があれば書込パルス発生器WPGより所定のタイミングで書込パルスWPが発生され、トラ

イステートバッファBF1～BF3が導通状態となり、命令IC_data、アドレスIF_PC_Lおよび有効性フラグinst_avail_inを書込ビット線WB上へ伝達する。この書込ビット線WB上のデータはそれぞれ選択されたエントリ（4つ）へそれぞれ同時に書込まれる。

書込んだデータを同時に読出すためには、書込パルスWPが発生され、書込ビット線WB上のデータが確定状態となった後に読出されることになるが、この場合、書込ビット線WB上に書込まれたデータが再びこのエントリ記憶装置を介してLラッチ回路14～16へ伝達されたあとにこれらのLラッチ回路がラッチ動作を実行する構成とされる。

また第26図および第27図に示す記憶素子はインバータラッチを用いており、このままでは、書込データと読出データが反転状態となることが考えられる。これは、トライステートバッファBF1～BF3をインバータ構成とすれば容易に対処することができ、またこのインバータは任意の箇所に設置することができる。

上述の構成により、アドレスqueue_topから始まる4つのアドレス領域に同時にそれぞれ異なるデータ（アドレス、命令および有効性フラグ）を書込むことができ、またアドレスscopeから始まる4つのアドレス領域に格納されたエントリの内容を同時に読出すことができる。

なお、上述の記憶装置においては、記憶素子としてインバータラッチ構成の記憶素子が用いられているが、この記憶素子の構成はどのようなものであってもよく、データを保持するものであればどのようなものであってもよい。

また、記憶装置の構成は上述のエントリ記憶装置の構成に限定されず、データ書込ポートおよび読出ポートが4つ設けられており、それぞれ隣接する4つのエントリ記憶装置の異なる書込/読出ポートが選択状態とされる構成であればどのような構成であってもよい。

さらに、上記実施例においては、命令およびアドレスはともに32ビットの場合および同時に読出される命令が4つの場合について説明したが、

これらのアドレスおよび命令長および同時にキューにおいて書込/読出される命令の数は任意の数であってもよい。

〔発明の効果〕

以上のように、この発明によれば、デコードステージにおいて、命令デコード前段に、命令メモリからの命令と、該命令の有効/無効を示すフラグと、該命令のアドレスとを1エントリとして複数の命令を同時に格納するキューを設け、命令メモリから同時に読出される複数の命令、およびアドレスならびに該命令の有効/無効を示すフラグを同時にキューに格納し、命令の実行状態、フェッチ状態等に従ってこのキューへの書込/読出を制御するように構成したため、命令デコードへ効率的に命令供給を行なうことができ、処理速度の速い並列処理装置を得ることができる。

4. 図面の簡単な説明

第1図はこの発明の一実施例である並列処理装置の命令供給ステージの構成を示す図である。第2図は第1図に示すキューの概念的構成を示す図

である。第3図はこの発明による並列処理装置におけるキューのアドレスの基本的な動きを示す図である。第4図はこの発明の並列処理装置におけるキューの書込および読出アドレスおよび有効性フラグの動きを示す図である。第5図はこの発明による並列処理装置において命令フェッチ要求が出された場合に命令供給が行なわれなかった場合のキューにおける書込および読出アドレスと有効性フラグの動きを示す図である。第6図は第1図に示すキュートップロジックの実現する論理を一覧にして示す図である。第7図は第1図に示すスコープネクストロジックの実現する論理を一覧にして示す図である。第8図は第1図に示す命令フェッチロジックの実現する論理を一覧にして示す図である。第9図は第1図に示す命令ミスロジックが実現する論理を一覧にして示す図である。第10A図は第1図に示すキューアペールロジックの実現する論理を一覧にして示す図である。第10B図は第10A図に含まれるキュー・フルの実現する論理動作を一覧にして示す図である。第1

0C図および第10D図は第10B図に示すキュー・フルの論理動作を説明するための図である。第11A図および第11B図はキューの第2の状態を説明するための図である。第12A図および第12B図はキューの第3の状態を説明するための図である。第13図は第1図に示すキューステイトロジックの実現する論理を一覧にして示す図である。第14図はキューの第1図に示す命令アペールロジックが実現する論理を一覧にして示す図である。第15図は第1図に示すキュー初期化状態ロジックが実現する論理を一覧にして示す図である。第16図は第1図に示すキューが基本的動きを行なう際の各ロジックの動作を示す信号波形図である。第17図は分岐発生時における第1図に示す各ロジックの動作を示す信号波形図である。第18図は命令フェッチ要求に対し命令供給が行なわれなかった場合の第1図に示す各ロジックの動作を示す信号波形図である。第19図は命令メモリからキューへ与えられる命令およびアドレスならびに有効性フラグの対応関係を概念的に

示す図である。第20図はキューへ与えられる命令の配置形態の一例を示す図である。第21図はキューへ与えられる有効アドレスの配置形態の一例を示す図である。第22図はキューへ与えられる有効性フラグの配置形態の一例を示す図である。第23図はキューの全体の構成を示す図である。第24図は第23図に示すエントリ記憶装置の構成を示す図である。第25図は第24図に示す命令記憶装置およびアドレス記憶装置の構成を示す図である。第26図は第25図に示す命令およびアドレス記憶装置および有効性フラグ記憶装置の構成の一例を示す図である。第27図は第26図に示すビット記憶装置の接続形態を例示する図である。第28図はキューのデータ読出時の動作を示す信号波形図である。第29図はキューのデータ書込時における動作を示す信号波形図である。第30図は並列処理装置の概念的構成を示す図である。第31図は、並列処理装置の一般的構成を示す図である。第32図は従来の並列処理装置における命令供給方法を示す図である。第33図は、

この発明が意図する命令供給方式を示す図である。
第 3 4 図は、第 3 3 図に示す命令供給方式を実現するために考えられることのできる命令供給方式を示す図である。第 3 5 図は第 3 4 図の命令供給方式を実現するための構成の一例を示す図である。第 3 6 図は並列処理装置において用いられる 2 相クロックを示す図である。

図において、1は命令メモリ、2は命令フェッチステージ、3は命令デコードステージ、9はキュー、30はキュートップロジック、31はスコープネクストロジック、32は命令フェッチロジック、33は命令ミスロジック、34はキューアペールロジック、35は命令アペールロジック、36はキュー初期化状態ロジック、50はキューステイトロジック、92-0~92-11はエントリ記憶装置、93-1は命令記憶装置、93-2はアドレス記憶装置、94は有効性フラグ記憶装置、95-0~95-31はビット記憶装置、MEは記憶素子、WRDは書込デコーダ、RDDは読出デコーダ、WPGは書込パルス発生器、B

F 1 ~ B F 3 はトライステートバッファである。

なお、図中、同一符号は同一または相当部分を示す。

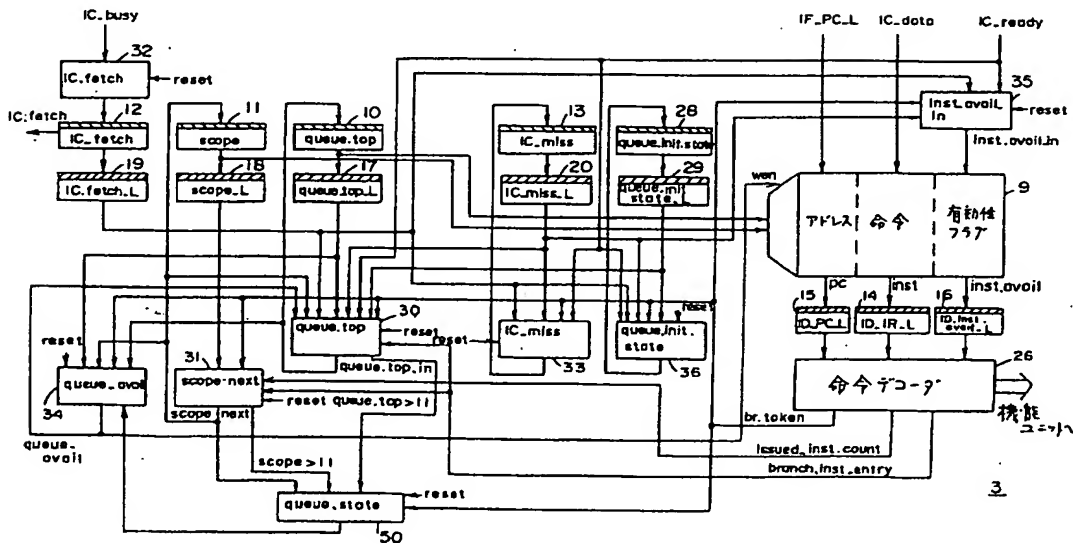
特許出願人 三菱電機株式会社

代理人 井理士 深見 久 郎

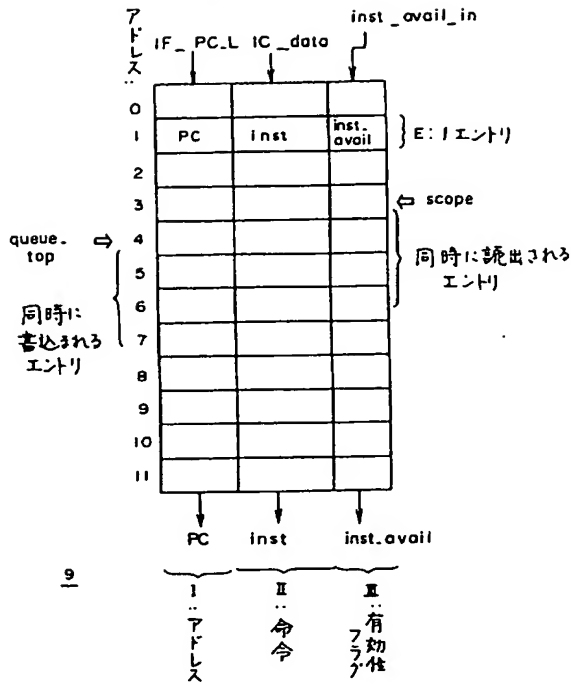
(ほか2名)



第 1 回



第2図



9

第4図

分岐が生じたときのキュー, queue_top, scopeの動き

キューのアドレス	0	1	2	3	4	5	6	7	8	9	10	11
(1) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												
(2) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												
(3) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												

第5図 プロセッサが次の命令フェッチ要求に対して、命令メモリから命令を読み出せない場合のキュー, queue_top, scopeの動き

キューのアドレス	0	1	2	3	4	5	6	7	8	9	10	11
(21) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												
(22) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												
(23) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												
(24) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												

第3図

キュー, queue_top, scopeの基本的な動き

キューのアドレス	0	1	2	3	4	5	6	7	8	9	10	11
(1) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												
(2) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												
(3) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												
(4) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												
(5) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												
(6) 命令有効性フラグ	1	1	1	1	1	1	1	1	1	1	1	1
queue_top												
scope												

x: 未定

第11A図

NORMAL 状態

キュー	命令	ポインタ
0		
1		
2	x	scope
3	x	
4	x	
5	x	
6		queue_top
7		
8		
9		
10		
11		

x: 未実行命令

第11B図

NORMAL 状態 / scope.next queue_top.in

キュー	命令	ポインタ
0		
1		
2		
3		
4		
5		scope.next
6		queue_top.in
7		
8		
9		
10		
11		

第12A図

REVERSE 状態

キュー	命令	ポインタ
0		
1	x	
2	x	queue_top
3		
4		
5		
6	x	scope
7	x	
8	x	
9	x	
10	x	
11	x	

x: 未実行命令

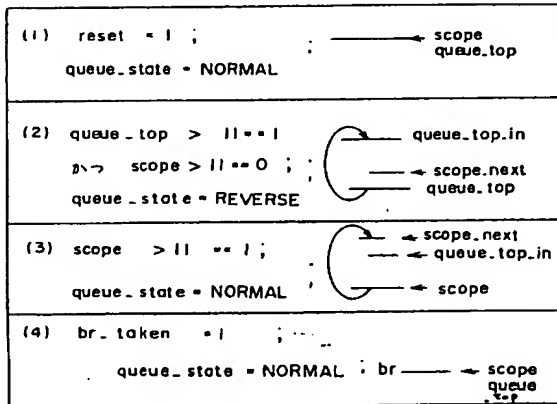
第12B図

REVERSE 状態 / scope.next queue_top.in

キュー	命令	ポインタ
0		
1	x	
2	x	
3	x	
4	x	
5	x	
6	x	scope.next
7		queue_top.in
8	x	
9	x	
10	x	
11	x	

x: 未実行命令

第13図



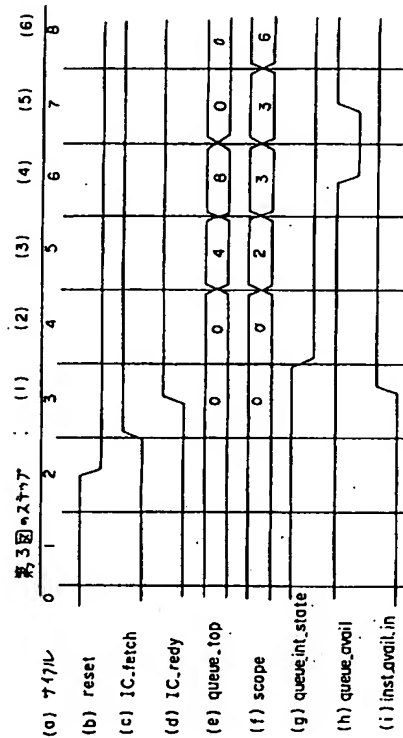
第14図

キューの「有効性フラグ」部分に書き込む値の論理

reset	br_taken	IC_fetch_L	IC_miss_L	inst.avail.in
1	x	x	x	0
0	1	x	x	0
0	0	1	x	IC_ready
0	0	x	1	IC_ready
0	0	0	0	0

x: 任意

第16図



キュー, queue_top, scope の基本的な動き

第15図

queue_init_state の論理
(ロジック 36)

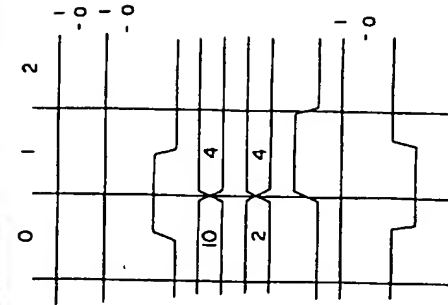
reset	br _taken	IC _fetch.L	IC _miss.L	IC _ready	queue_init_state
1	x	x	x	x	1
0	1	x	x	x	1
0	0	1	x	0	1
0	0	1	x	1	0
0	0	0	x	1	1
0	0	0	1	0	1
0	0	0	0	1	0
0	0	0	0	0	x
0	0	0	0	0	queue_init_state.L

x: 任意

第17図

分岐発生時のキュー、queue_top、scopeの動き

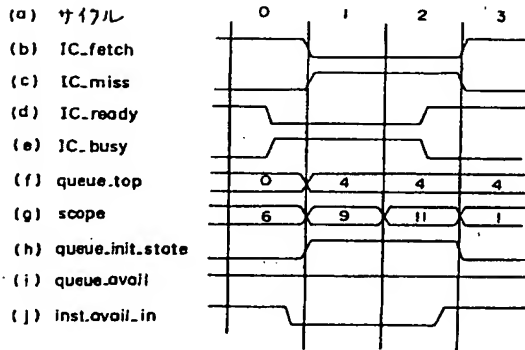
(11) (12) (13) 第4図のステップ



- (a) サイクル
- (b) IC_fetch
- (c) IC_ready
- (d) br_taken
- (e) queue_top
- (f) scope
- (g) queue_init_state
- (h) queue_avail
- (i) inst_avail_in

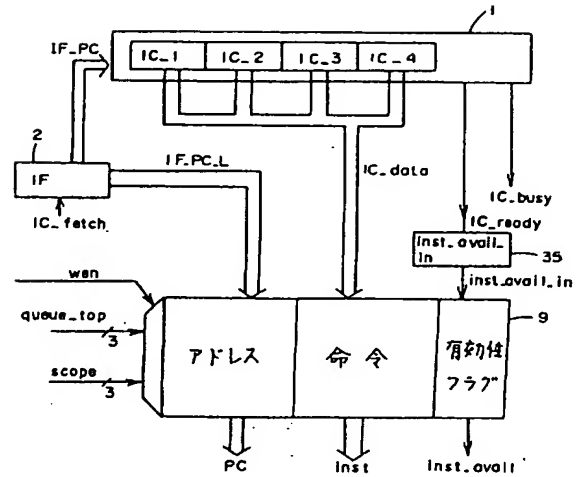
第18図

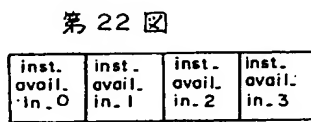
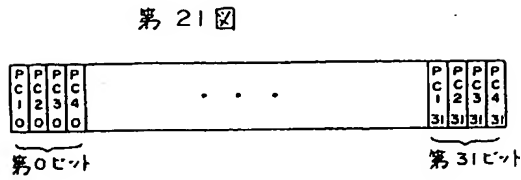
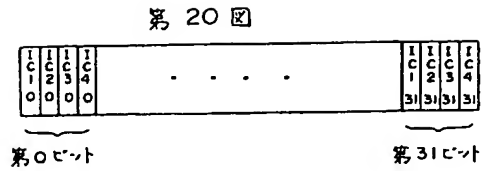
第5図のステップ: (21) (22) (23) (24)



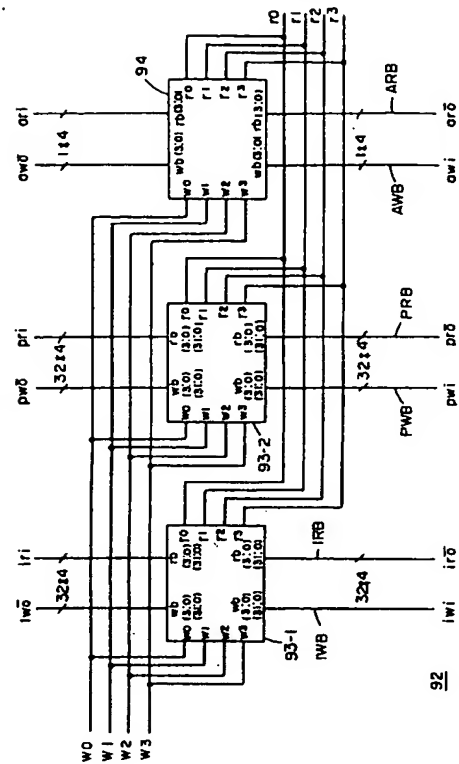
発生された命令フェッチ要求に対して命令メモリから命令が供給されない場合のキュー、queue_top および scope の動き

第19図

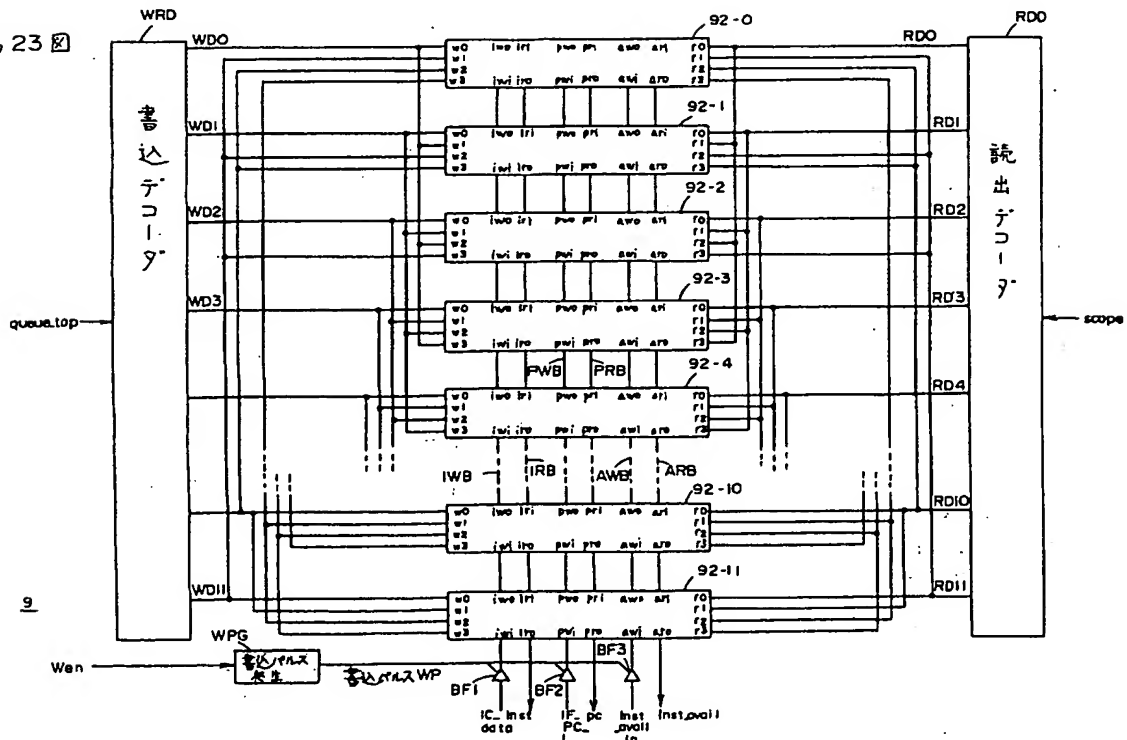




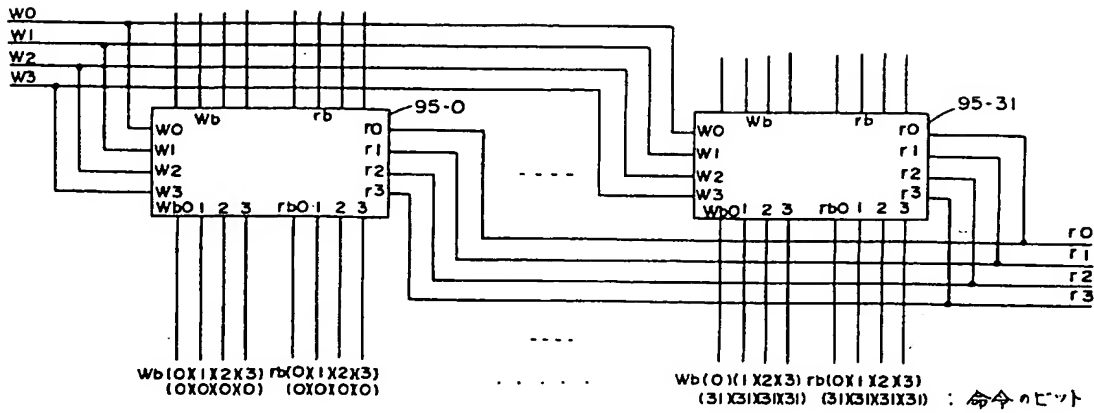
第24図



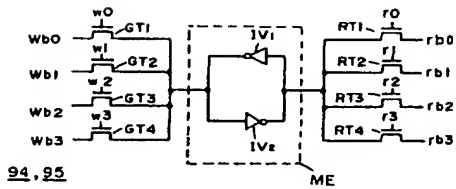
第23図



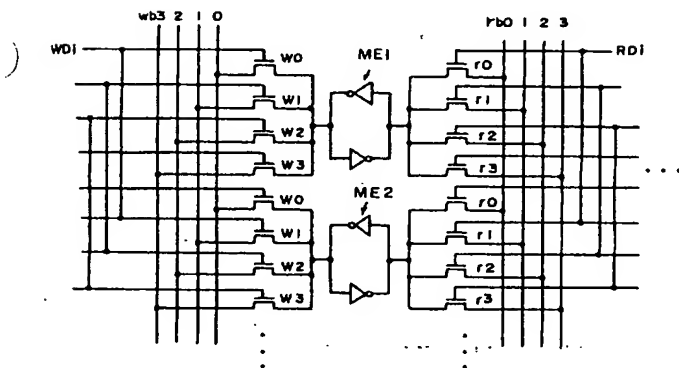
第 25 図



第 26 図

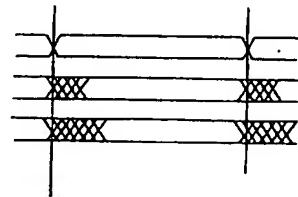


第 27 図



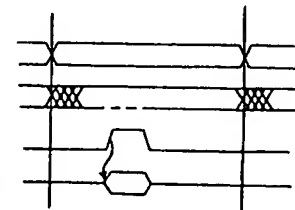
第 28 図

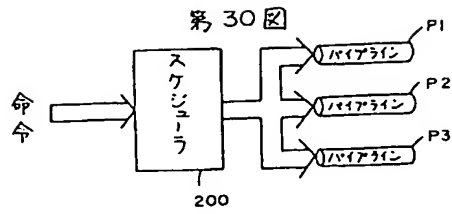
- (a) scope
- (b) 読出しワード線 RD
- (c) 読出しビット線 RB



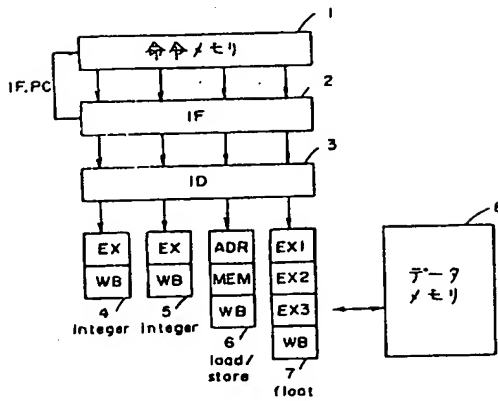
第 29 図

- (a) queue_top
- (b) 書き込みワード線 WD
- (c) 書き込みパルス wp
- (d) 書き込みビット線 WB





第31図



第32図

サイクル	命令
1	[1] 2 3 4 ; 1の発行
2	[2] [3] 4 ; 2,3の発行
3	[4] ; 4の発行
4	[5] [6] 7 8 ; 5,6の発行
5	[7] [8] ; 7,8の発行
6	...

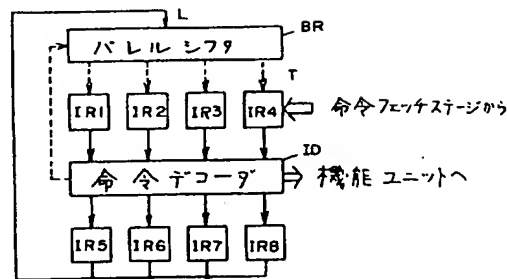
第33図

サイクル	命令
1	[1] 2 3 4 ; 1の発行
2	[2] [3] 4 5 ; 2,3の発行
3	[4] [5] [6] 7 ; 4,5,6の発行
4	[7] [8] 9 10 ; 7,8の発行
5	...

第34図

- (1) [2] [3] 4 5 ; 命令2,3の発行
 (2) 4 5 - - ; 命令レジスタのシフト
 (3) 4 5 6 7 ; 空いた命令レジスタへの命令のフェッチ

第35図



第36図

